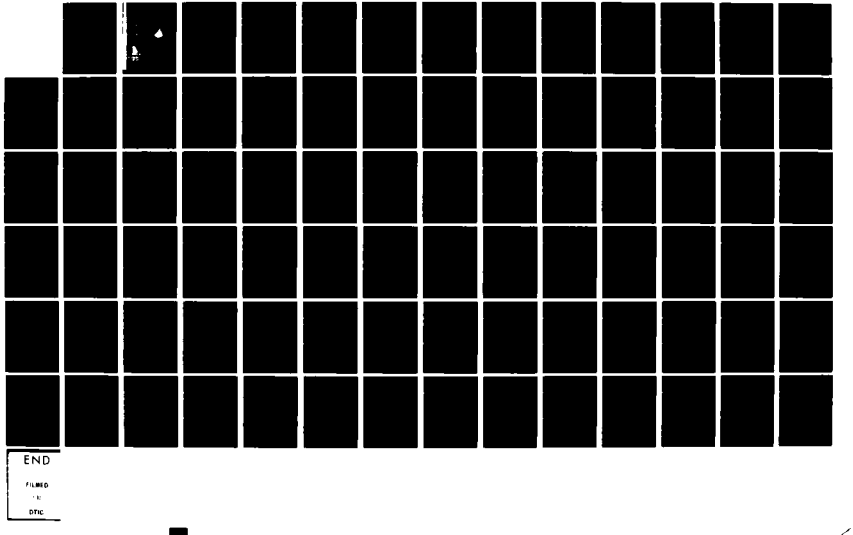
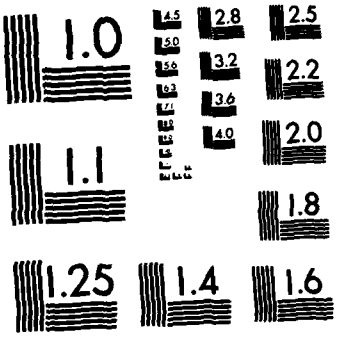
AD-A122 813 AGGREGATE NETWORK MODEL OF SHIP OVERHAUL(U) CALIFORNIA 1/1
UNIV BERKELEY OPERATIONS RESEARCH CENTER
R C LEACHMAN ET AL. MAR 82 ORC-82-2 N00014-76-C-0134
UNCLASSIFIED F/G 13/10 NL



END

FILMED

DTIC

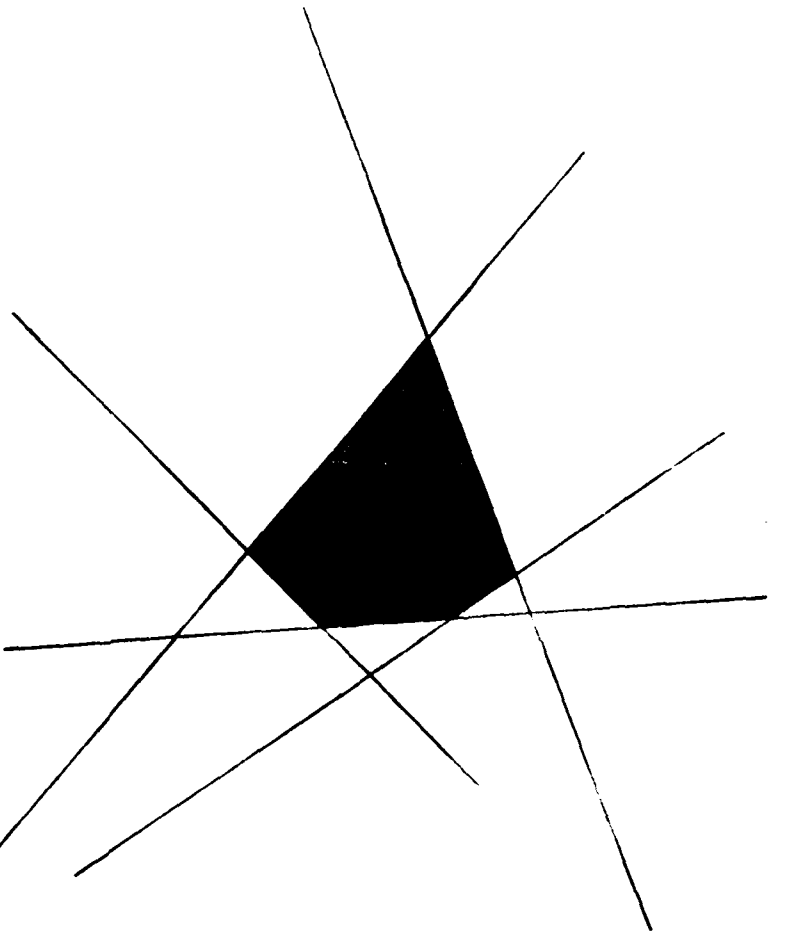


MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(12)

AGGREGATE NETWORK MODEL OF SHIP OVERHAUL

by
ROBERT C. LEACHMAN
and
JOERG BOYSEN



OPERATIONS RESEARCH CENTER

UNIVERSITY OF CALIFORNIA · BERKELEY

AGGREGATE NETWORK MODEL OF SHIP OVERHAUL

by

Robert C. Leachman
Department of Industrial Engineering
and Operations Research
University of California, Berkeley

and

Joerg Boysen
Operations Research Center
University of California, Berkeley

MARCH 1982 ORC 82-2

This research was supported by the Office of Naval Research under Contract N00014-76-C-0134 with the University of California. Reproduction in whole or in part is permitted for any purpose of the United States Government.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ORC 82-2	2. GOVT ACCESSION NO. A122 813	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* AGGREGATE NETWORK MODEL OF SHIP OVERHAUL		5. TYPE OF REPORT & PERIOD COVERED Research Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Robert C. Leachman and Joerg Boysen		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0134
9. PERFORMING ORGANIZATION NAME AND ADDRESS Operations Research Center University of California Berkeley, California 94720		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR 047 033
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Department of the Navy Arlington, Virginia 22217		12. REPORT DATE March 1982 13. NUMBER OF PAGES 81
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Multiproject Planning
Linear Programming
Resource Allocation
Aggregate Planning
Activity Networks
Capacity Planning
Construction Systems

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

(SEE ABSTRACT)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

Abstract

A new approach is presented for the problem of allocating labor resources among ship overhaul projects in a shipyard. It is based on a management-level model of project execution, which describes the alternative distributions of labor use over the life of a project. The model consists of a network of major project components called aggregate activities. It includes mathematical constraints which ensure that activities operate consistently with each other. The analysis of data on a large overhaul provided valuable insight for construction of the model.

A linear program (LP) is used to dynamically allocate labor resource capacities among projects. The LP formulation allows the impact of alternative allocations and capacities to be studied. Plans for future projects can be evaluated under various assumptions. Problems of realistic size are readily handled by existing computer packages.

The aggregate model of project execution is proposed as an effective tool for planning of multi-project resource use. Although the model was developed in the context of shipyard planning, it is believed to be equally applicable to other repetitive construction industries, for example, aircraft or radar systems manufacturing. It provides a link between resource planning and detailed project planning. As such, it can improve management's control over project performance and labor productivity.

TABLE OF CONTENTS

LIST OF EXHIBITS

GLOSSARY OF NOTATION

A_i	aggregate activity, $i=1,\ldots,N$ (3.2.1)
a_i	total labor use by A_i (3.2.1)
a_{ki}	amount of R_k used by A_i per unit intensity of A_i (3.1,3.2.1)
$\bar{a}_{hi}$	amount of IP_h used by A_i per unit intensity of A_i (3.1,3.4.1)
b_l	total labor use by KO_l (2.4.2)
b_{kl}	amount of R_k used by KO_l (2.4.2)
c_{hi}	amount of IP_h produced by A_i per unit intensity of A_i (3.1,3.4.1)
D_l	duration of key-op KO_l (2.4.1)
ES	early key-op start time schedule, $ES=\{ES_1,\ldots,ES_L\}$ (2.4.1)
ES_l, EF_l	early start and finish time of KO_l (2.4.1)
F_i	upper bound on operation of activity A_i (3.2.1)
F_i^m	m^{th} partitioning point of window of A_i (3.4.3)
F_l	finish time of key-op KO_l (2.4.1)
$I_i(t)$	proportion of each intermediate product used by A_i up to t (3.4.1)
IP_h	intermediate product, $h=1,\ldots,H$ (3.1,3.4.1)
KO_l	key-op (key operation), $l=1,\ldots,L$ (2.2)
LS	late key-op start time schedule, $LS=\{LS_1,\ldots,LS_L\}$ (2.4.1)
LS_l, LF_l	late start and finish time of KO_l (2.4.1)
$[m]_i$	m^{th} subinterval of window of A_i, $[m]_i=(F_i^{m-1},F_i^m]$, $m=1,\ldots,M$ (3.4.3)
$O_i(t)$	proportion of each intermediate product produced by A_i up to t (3.4.1)
OH_j	overhaul, $j=1,\ldots,J$ (4.2.1)
OM_i	operating mode of activity A_i, $OM_i=\{Z_i(t)\}_t$ (3.2.1)
OM_i^E, OM_i^L	early and late operating modes of A_i (3.2.4)
OM_j^*	tight operating mode of A_j with its predecessors (3.4.3)
$\hat{OM}_i^E$	attainable early operating mode of A_i (3.5.1)
P_i	earliness of operation of A_i (4.2)
$p_i(t)$	relative progress of A_i at t (3.4.3)
$p_i[1], p_i[3]$	relative progress of A_i in first and last phase of window (3.7.1)
R_k	labor resource, $k=1,\ldots,K$ (2.4.2)

S	key-op start time schedule, $S=\{S_1, \ldots, S_L\}$ (2.4.1)
S_i	lower bound on operation of activity A_i (3.2.1)
S_l	start time of key-op KO_l (2.4.1)
$(S_i, F_i]$	operating window of A_i (3.2.1)
T	time horizon (3.1,4.2)
$X(t)$	total overhaul labor use by time t (4.4)
$X^E(t), X^L(t)$	total overhaul labor use by time t, given early or late execution (4.4)
$\overline{X}(t)$	cumulative total labor shop capacity (4.4)
$x_{ki}(t)$	amount of R_k used during $(t-1, t]$ by A_i (3.2.2)
$x_{ki}^S(t)$	amount of R_k used by A_i during $(t-1, t]$, given S (3.2.2)
$x_{kl}^S(t)$	amount of R_k used by KO_l during $(t-1, t]$, given S (2.4.2)
$x_k^j(t)$	amount of R_k used by overhaul OH_j during $(t-1, t]$ (4.2.1)
$\{x_k^j(t)\}_t$	allocation of R_k to OH_j (4.2.1)
$Z_i(t)$	cumulative intensity of A_i at time t (3.2.1)
$Z_i^E(t), Z_i^L(t)$	cumulative intensity of early and late operation of A_i at t (3.2.4)
$\hat{Z}_i^E(t)$	attainable early cumulative intensity of A_i at t (3.5.1)
$z_i(t)$	intensity of A_i during $(t-1, t]$ (3.1)
$z_i^S(t)$	intensity of A_i during $(t-1, t]$, given S (3.2.2)
$z_i[2]$	intensity in middle phase of window of A_i (3.7.1)
$\overline{z}_i(t)$	intensity bound of A_i during $(t-1, t]$ (3.2.3)
$\hat{z}_i$	workspace limitation of A_i (3.5.1)
$\overline{z}_l$	operating intensity of KO_l (2.4.2)
α_t	objective function coefficient (4.2)
$\Gamma(S, t)$	index set of key-ops within A_i that start by t, given S (3.4.2)
$\Delta(S, t)$	index set of key-ops within A_i that finish by t, given S (3.4.2)
$\Lambda(S, t)$	index set of key-ops within A_i that operate at t, given S (3.2.2)
σ_i	slack of aggregate activity A_i (3.2.4)
$\sigma_i(t)$	component of σ_i "accruing" from time period t (4.2)
σ_l	slack of key-op KO_l (2.4.1)

1. INTRODUCTION TO NAVAL SHIPYARD PLANNING AND SCHEDULING

1.1. Current Practice

Production activity in a naval shipyard consists of a variety of large ship overhaul projects executed concurrently. Each of these projects places time-varying workloads on 20 or more distinct labor trade shops, with total labor input to each project in the tens or even hundreds of thousands of man-days. Project durations range from 3 months to 3 years.

Overhaul projects must be phased in time to ensure the availability of certain key facilities such as dry docks. Furthermore, the projects should be phased and scheduled so as not to overload any of the trade shops. However, the phasing should not be so much that shops are underloaded, in which case workers tend to slow down. In the event that a shop is underloaded, productivity is diminished, and hence overhaul costs are increased. The trade-off of assuring schedule adherence versus maintaining high productivity is a complex management problem.

To motivate the research reported in this paper, a brief summary of the current practice of planning and scheduling in naval shipyards is presented here. (This summary is an expansion of the one provided in [7].) In current practice, a two-stage procedure is utilized for overhaul planning and scheduling.

First, an aggregate planning effort is undertaken. Starting, ending, and several intermediate *key event* (milestone) dates are established for each project. These milestones mark the realization of key phases of the overhaul, such as docking, undocking, test completions, etc. The goal is to select dates for each overhaul which are feasible but which also promote efficient use of yard resources.

Second, a detailed activity network is developed for each overhaul. Estimates of duration and required shop man-days are developed for each such activity, and the activities are scheduled consistently with the pre-established target key event dates. The goal of this effort is to schedule and control the execution of each overhaul, so that milestone dates established in aggregate planning may be realized.

Aggregate planning is accomplished using an iterative tabular/graphical technique known as "planning on the curve." In this method, a file of standardized loading curves are maintained for each class of ship. These curves portray the percentages of total man-days of labor expended in each month of an overhaul, given proposed durations for the realization of

Month	Percentage of Total Labor Expended	Cumulative Percentage
1	1.4	1.4
2	5.9	7.3
3	12.3	19.6
4	15.8	35.4
5	16.3	51.7
6	14.2	65.9
7	11.4	77.3
8	8.9	86.2
9	6.4	92.6
10	4.3	96.9
11	2.4	100.0

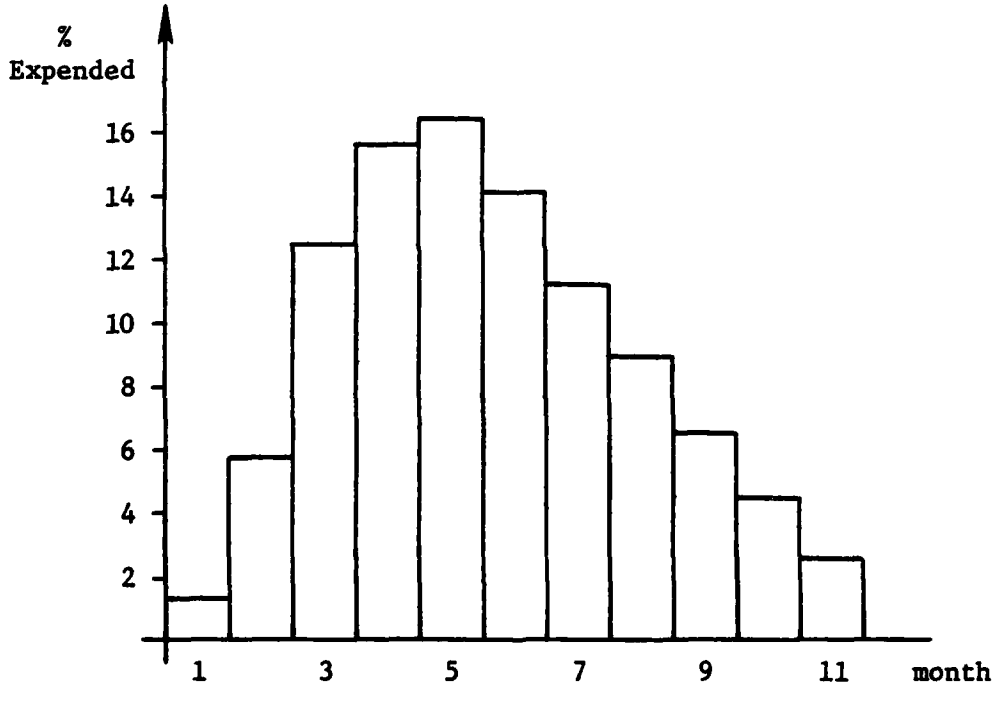


EXHIBIT 1.1

EXAMPLE OF STANDARDIZED LOADING CURVES

overhaul milestones. See Exhibit 1.1 for an example. By multiplying the total man-day estimate for an overhaul times the percentages, an estimated month-by-month time history of total labor workload for an overhaul is derived.

If start dates and durations for each future and ongoing overhaul are specified, the overhaul time histories may be summed in tabular form to develop an estimated total man-day workload history for the shipyard. The resulting curve is compared to a time history of yard "capacity", i.e., a time history of the total productive work force, as measured in man-days per month. See Exhibit 1.2 for an example.

As shown in the exhibit, for particular trial overhaul durations and start dates, the variance between the workforce and workload curves may be both positive and negative. A planner would experiment by trial and error with different overhaul start times and durations in an attempt to match the workload curve and the workforce curve reasonably closely.

Recall that the different labor shops have been aggregated, so that an aggregate plan in which total workload coincides with total workforce may still be characterized by overloading and underloading of particular shops. Costs of underloading are thought to be more severe than costs of overloading, so that in practice an aggregate plan is sought in which the workload curve lies about 10% above the workforce curve.

Each time an aggregate plan is completed, key event dates are fixed for all upcoming overhauls which will begin within some scheduling lead time; dates for other overhauls are treated as tentative, and may be changed in future plans.

Once milestone dates for an overhaul have been set, the scheduling of detailed, component activities of the overhaul is undertaken. Typically, thousands of activities known as *key-ops* are defined, an undertaking requiring a year or more. Durations and man-hour requirements from each trade shop are estimated for each key-op, and a critical path network of the key-ops is developed. In the scheduling effort, network slack between milestone dates is uniformly allocated, which tends to reduce shop loading peaks.

Milestone dates and key-op schedules are distributed to line (shop) production management, which is responsible for the day-to-day allocation of working crews and job supervision. In practice, adherence to key-op schedules is rare, but observance of milestone dates is accomplished where feasible.

PROJECTS	MONTHLY LOADS (man-day totals)						
	FEB	MAR	APR	MAY	JUN	JUL	AUG
PROJECT A	2031	1648	1289	1048	936	854	821
PROJECT B	2059	1896	1182	856	734	600	520
PROJECT C	1488	1759	2126	2021	1198	514	378
PROJECT D			234	1175	1860	2432	2914
OTHER	410	390	380	370	370	385	400
TOTAL LOAD	5988	5693	5211	5470	5098	4785	5033
WORKFORCE	4975	4906	4810	4863	4915	4873	4894
OVERLOAD (UNDERLOAD)	1013	787	401	607	183	(88)	139

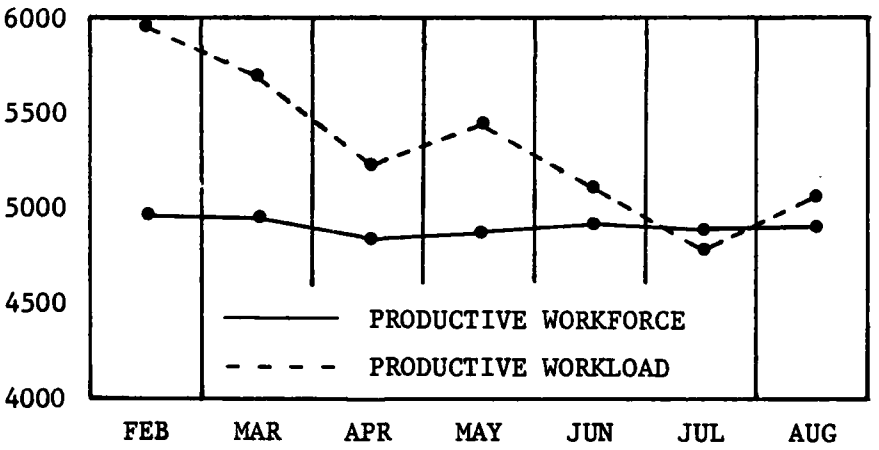


EXHIBIT 1.2

MULTI-OVERHAUL SCHEDULING CHART

1.2. A System for Improved Planning and Scheduling

In our opinion, there are several problems with this method of planning and scheduling which we feel are worthy of research and development. As previously mentioned, aggregate planning is performed by aggregating capacities of non-substitutable resources. If a method for planning were developed in which trade shops were treated as distinct resources, the underloading and overloading of shops inherent in the current method could be reduced, providing the potential for gains in both productivity and schedule adherence.

Second, linkage between aggregate planning and project scheduling now exists only in the form of milestone dates. The aggregate plan is prepared with certain resource load histories in mind, but there is no guarantee that loading histories of detailed overhaul schedules are consistent with aggregate plans. The appropriate allocation of shop capacities to an overhaul is only implicitly suggested by milestone dates.

Finally, overhaul scheduling and control is attempted at too fine a level of detail, since key-op schedules are largely ignored in actual operations. Less detailed key operations need to be defined, in which the level of detail is more consistent with the scheduling constraints perceived by line management. Networks with hundreds of key-ops rather than thousands should be used for scheduling and control.

We envision an ideal shipyard planning and scheduling system as shown in Exhibit 1.3. A more detailed, computer-aided aggregate planning effort would be undertaken which explicitly considers the various trade shop capacities. Key event dates *and* allocations of shop capacities would be established for each overhaul based on a plan which is feasible and maximizes yard productivity. In parallel with aggregate planning, detailed overhaul planning would take place, in which standardized key-op networks appropriate for scheduling and control are calibrated as to resource requirements and duration. Overhaul scheduling and control would be conducted using such data. Moreover, schedules would be derived which are consistent with the key event and shop capacity constraints specified in aggregate plans. The execution of the overhaul would be controlled by monitoring the completion times of scheduling activities, which we term *principal events.*

This paper concerns the development of an aggregate planning methodology for such a system. Central to the methodology is the development of an aggregate overhaul model, with (aggregate) activities defined at the *key event* level of detail. This model builds on the dynamic production modeling work of Shephard et al. [10] and Leachman [6], but work flow between

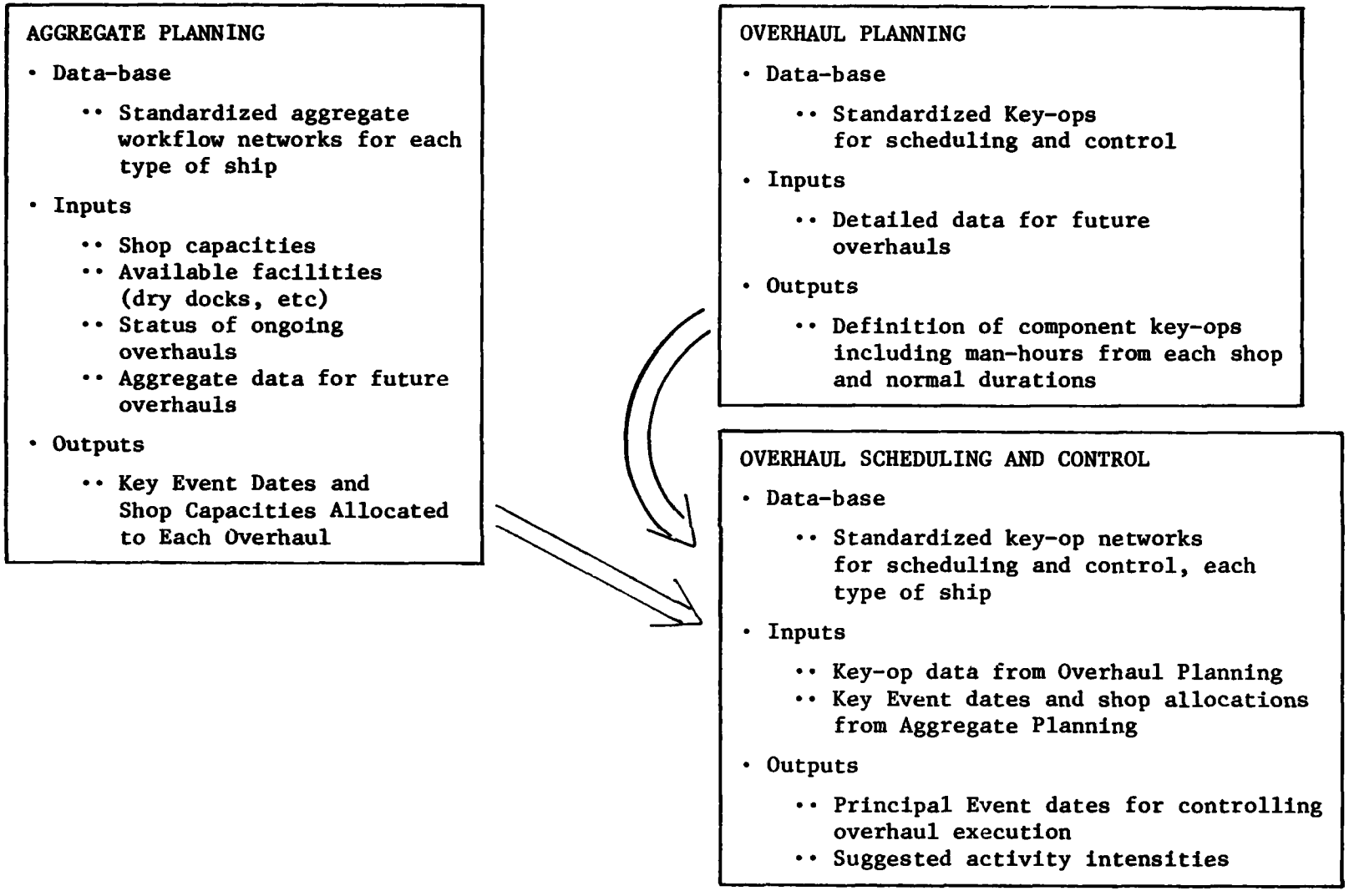


EXHIBIT 1.3

IDEAL SHIPYARD PLANNING AND SCHEDULING SYSTEM

activities (intermediate product transfer) is treated in a new manner.

The reader is also referred to [7] and [11]. In [7], the aggregation of a 1150 key-op network into a 430 key-op network (at the principal event level of detail) is discussed, as well as multi-shop resource leveling techniques for the derived network. In [11], scheduling of the same network subject to shop capacity allocations is investigated. At this point we terminate our general discussion of the planning and scheduling process in naval shipyards, and proceed to the technical development of new models appropriate for aggregate planning.

2. NETWORK REPRESENTATION OF SHIP OVERHAULS

2.1. Introduction to Activity Networks

A construction project can be represented graphically as a collection of *activities* which are the nodes of a *network*. The arcs of the network indicate *dependence relationships* between pairs of activities, as described below.

Each activity operates by using resources (labor from different shops) over a period of time. These are denoted by $R_1, \ldots, R_K$. Its operation also depends on the availability of intermediate products from other activities (its *predecessors*). The activity itself produces intermediate products that other activities (its *successors*) use. The network arcs refer to these dependences. The generalization of network arcs to represent product transfers rather than simple precedences is due to Shephard et al [10].

Transfers of intermediate products between activities may consist of flows of work in process, or may effectively be "go-ahead" signals. In detailed networks an activity produces a distinct product for each of its successors; in more aggregate networks there may be "sharing" or "pooling" of intermediate products, as will be discussed in Chapter 3.

In this chapter we will encounter two types of activity networks. The *key-op network* is a detailed representation of a ship overhaul. The *aggregate activity network* is a management-level representation with less detail. They differ with respect to assumptions about activity operation and intermediate product transfer between activities.

We have chosen the *activity-on-node* representation of an activity network since it emphasizes the flow of intermediate products through the network. The equivalent, *activity-on-arc* networking scheme is also frequently used. (See, for example, Moder and Phillips [8].)

2.2. Key-op Networks

For each overhaul, planning information exists in considerable detail. Much of it is based on previous, similar overhauls. Hundreds or even thousands of detailed activities, called *key-ops* (key operations), are defined for overhaul scheduling and cost accounting purposes. For each key-op, estimates are prepared of work duration and labor use by trade shop. A key-op usually requires labor from several shops with one, the *lead shop,* being dominant.

The dependence relationships between key-ops are described in the *key-op network.* Traditional scheduling assumptions require the dependences to be *strict precedences:* a key-op may

start only after its predecessors have finished. For this reason, key-ops are defined at a level of detail at which transfer of intermediate products is *event-based*, i.e., when the key-ops finish. A distinct product is then transferred to each successor key-op.

The decomposition of an overhaul into key-ops is guided by four main criteria:

- precedence requirements (as just described)
- labor shop emphasis ("lead shop")
- work location
- technical system.

Key-ops are defined at a level of detail such that all of these work breakdown criteria are applied. Leachman and Boysen present examples of key-ops in Reference [7].

2.3. Key Events

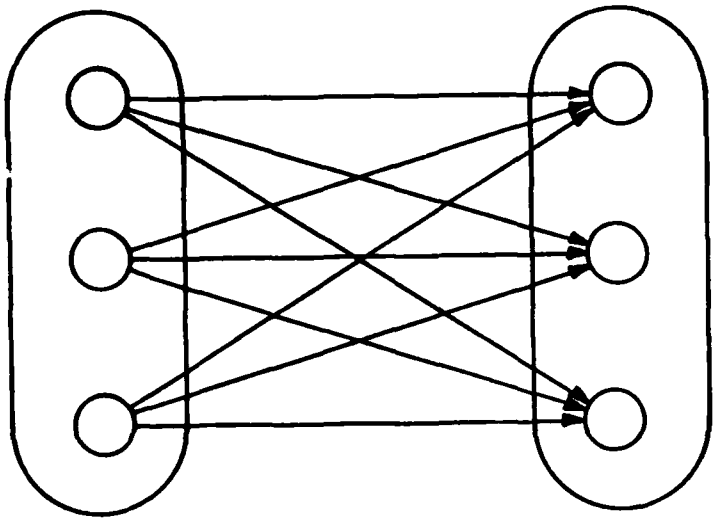
The completion of each key-op is an event. To control the execution of an overhaul, the shipyard identifies certain *key events*, which represent the completion of logical phases of the overhaul or the realization of certain work accumulations. We distinguish two types of key events: restraining events and monitoring events, as discussed below.

Restraining Events

Referring to Exhibit 2.1(a), suppose a group A of key-ops must finish before key-ops in another group B can start. We could represent this in the key-op network by connecting each of the key-ops in group A to all of those in B. A better and clearer way is to define a *key event node*, represented by a square, which succeeds key-ops in A and precedes key-ops in B, as in Exhibit 2.1(b). The key event node *restrains* group B key-ops until group A has finished.

This type of restraint also arises when certain events are fixed in time: group A key-ops must operate before the event, and group B key-ops afterwards. In either of the above cases, the restraining event is a control point for transfer of intermediate product between the groups. Of course, event nodes in the network use no resources and have no duration. Several important restraining events occur during the execution of an overhaul, for example:

- docking and undocking
- completion of insulation removal before machinery space components can be removed
- completion of all component testing before testing of a whole technical system may begin.



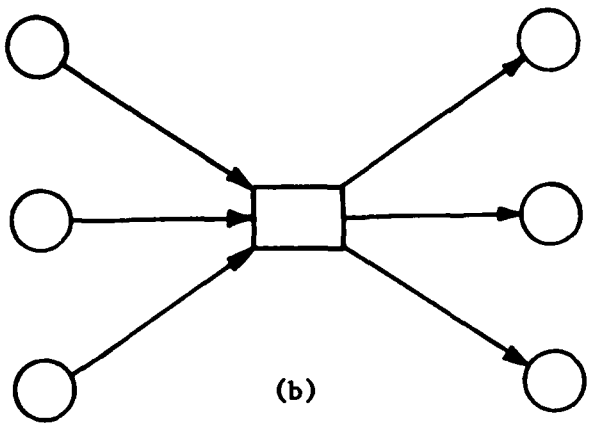


EXHIBIT 2.1

REPRESENTATIONS OF A RESTRAINING EVENT

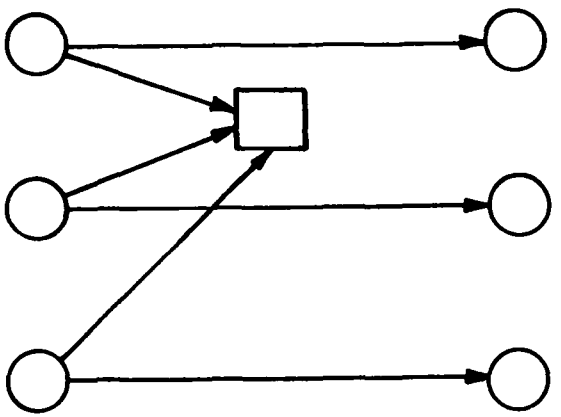


EXHIBIT 2.2

MONITORING EVENT NODE

Monitoring Events

Referring to Exhibit 2.2, this type of event is defined when the completion of a group A of key-ops is to be *monitored.* We represent it by connecting the key-ops in group A to an event node. Note that the monitoring event does not restrain the start of the successors of key-ops in group A.

As an example, consider the event "Completion of all Shop Repairs to Engine Room Components." After each component is repaired it can be reinstalled immediately. But to monitor the completion of all repairs, or equivalently, the *transfer* of all repaired components to the ship, this event is defined. As in the case of restraining events, monitoring events are control points for product transfer with no resource use or duration.

Key Event Networks

The key events can be linked together in the form of a network. This *key event network* is not an activity network; rather, it displays sequences of key events and possibly their relative timing. It can be thought of as a skeletal structure for the key-op network. The potential operating periods for key-ops can be constrained by specification of dates for key events.

The key event network provides a way of summarizing the execution of an overhaul. In Section 2.5 we propose a different kind of summary network which focuses on major overhaul *activities* rather than *events.* In Section 3.3.1 the connection between the two is discussed.

2.4. Analysis of Key-op Networks

In this section techniques for scheduling of key-ops are briefly reviewed. In later sections we will refer to the notation used here. We denote the key-ops within an overhaul by $KO_1, \ldots, KO_L$.

2.4.1. Critical Path Method

Each key-op KO_l is assumed to have a *duration* D_l. Let $S=\{S_1, \ldots, S_L\}$ be a *schedule* of key-op *start times.* This schedule, together with the key-op durations, describes the execution of the overhaul. Given a start time S_l for key-op KO_l, its *finish time* F_l is simply $F_l \equiv S_l + D_l$. We assume that KO_l operates in the half-open interval $(S_l, F_l]$, and that S_l and D_l are expressed in integral time units.

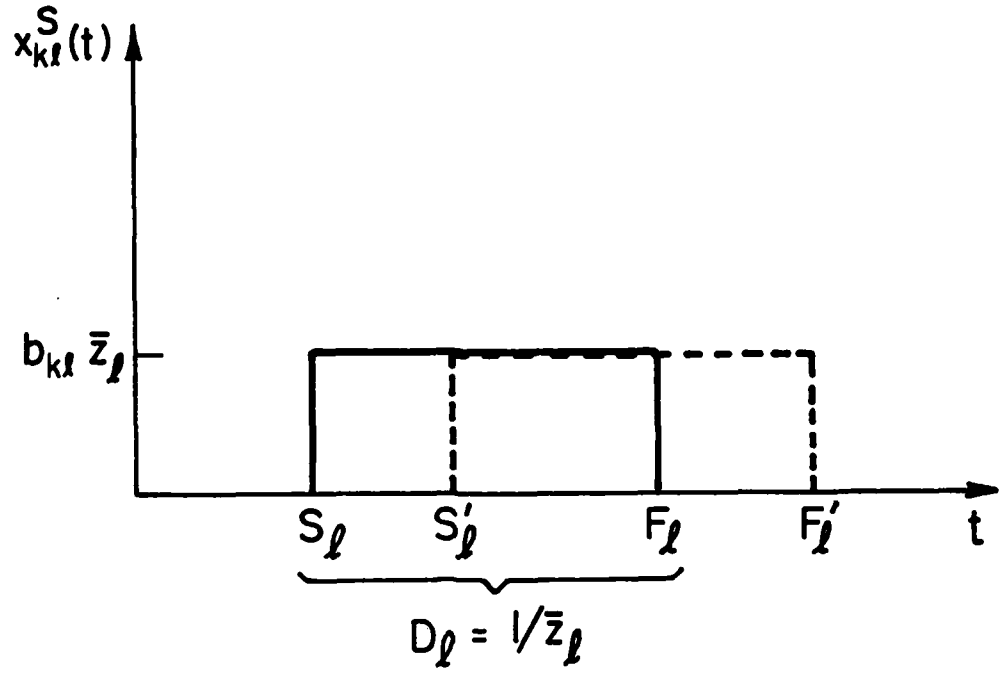


EXHIBIT 2.3

SHIFTING KEY-OP RESOURCE LOADS

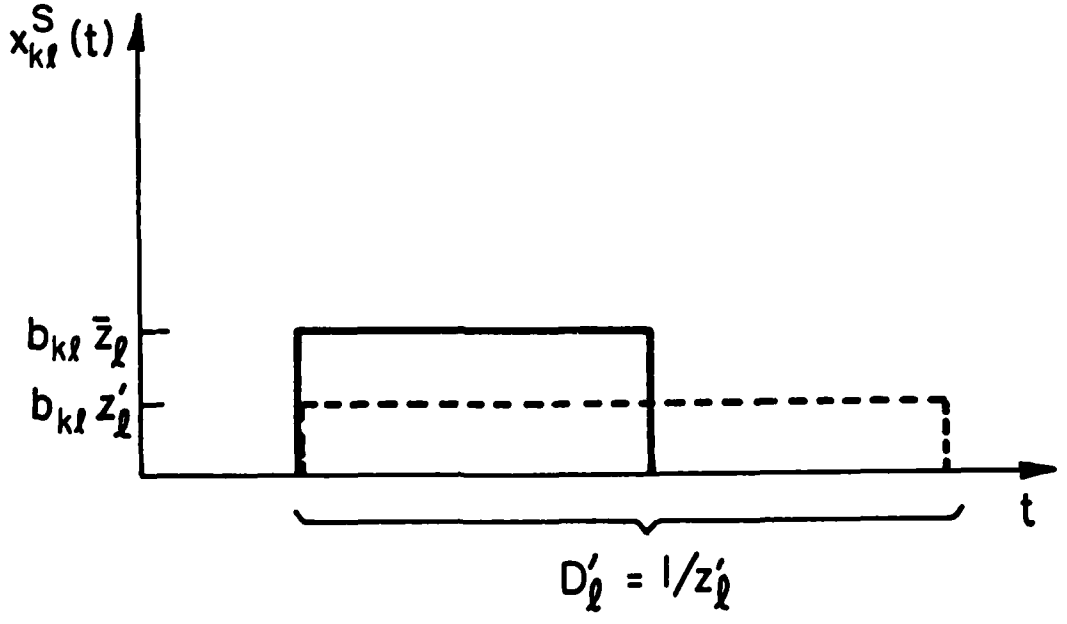


EXHIBIT 2.4

STRETCHING KEY-OP RESOURCE LOADS

The Critical Path Method (CPM) is a well-known technique for determining the set of *feasible* start schedules. (See Moder and Phillips [8], Harris [3], Elmaghraby [2].) Given overhaul start and finish times, and possibly key event dates, we can bound the start time of each key-op in the following way. Let KO_l be a key-op with a successor KO_m. Then KO_l must finish before KO_m can start, i.e.,

$$F_l \leqslant S_m \Leftrightarrow S_l + D_l \leqslant S_m \ . \tag{2.1}$$

Given all start time constraints of type (2.1), we can compute the *earliest* and *latest* possible start times ES_l and LS_l for each key-op KO_l, and the corresponding finish times EF_l and LF_l. Every feasible start schedule $S=\{S_1, \ldots, S_L\}$ thus satisfies

$$ES_l \leqslant S_l \leqslant LS_l \qquad l=1,\ldots,L \ .$$

The *early schedule* is denoted by $ES=\{ES_1, \ldots, ES_L\}$ and the *late schedule* by $LS=\{LS_1, \ldots, LS_L\}$.

The *slack* (or float) of key-op KO_l is defined as $\sigma_l \equiv LS_l - ES_l$. It is the amount of time that the start of KO_l can be delayed beyond ES_l if its successors operate late and its predecessors early. The key-ops with zero slack form the *critical paths* of the network.

2.4.2. Resource Use Analysis

It is assumed that each key-op KO_l uses resources at a constant rate. Let b_{kl} be the amount of resource (labor shop) R_k that KO_l uses. The *operating intensity* $\bar{z}_l$ is defined as the proportion of total resource requirement of KO_l that is used per unit of time, so that $\bar{z}_l \equiv 1/D_l$.

Given these assumptions on key-op resource use, a start schedule S completely determines resource use during the execution of the overhaul. Let $x_{kl}^S(t)$ be the amount of resource R_k used by KO_l at time t. Then

$$x_{kl}^S(t) = \begin{cases} b_{kl}\bar{z}_l & \text{if } S_l < t \leqslant F_l \\ 0 & \textit{otherwise} \end{cases} \ . \tag{2.2}$$

Exhibit 2.3 graphs this resource load profile.

Various techniques for resource-constrained project scheduling exist; see, for example, Herroelen [4], Cooper [1], Patterson [9]. Most methods employ heuristic rules of shifting

activities (key-ops) forward or backward in time, with the objective of keeping total resource use within the given constraints. This corresponds to shifting resource load "boxes" in time, as Exhibit 2.3 illustrates. Leachman [5] proposes a more flexible scheme in which key-op intensity z_l, while constant over time, is a decision variable. This is equivalent to allowing different key-op durations z_l^{-1}, and stretching or compressing the resource load boxes accordingly, as in Exhibit 2.4.

2.5. Aggregate Activity Networks

We are interested in representing the execution of an overhaul at a level of detail that facilitates allocation of shipyard resources among overhauls. The key-op network is not appropriate for this purpose for two reasons:

(1) Data Generation: A comprehensive decision model that attempts to set key-op schedules for each overhaul while determining resource allocations to overhauls would require huge amounts of data. These data are not always available when they would be needed.

(2) Computational Problems: Even if all data were available, the event-based restrictions on resource use implied by the key-op network* make techniques of resource allocation computationally complex. Performing the scheduling simultaneously for all overhauls becomes impractical.

For these reasons a less-detailed model of overhaul execution was developed. We introduce the concept of an *aggregate activity network.* Each aggregate activity is a collection of many key-ops. The network arcs indicate transfer of intermediate product, as before. Product transfer is no longer event-based, however, as we see below.

We would like to define the activities at the highest degree of aggregation that still permits a reasonable representation of resource use. Since the level of detail is reduced, we cannot decompose the overhaul tasks according to all of the criteria of Section 2.2, at least not to the same extent. Recall that these criteria were as follows:

- precedence requirements
- labor shop emphasis
- work location
- technical system.

*a key-op can start to use resources only after its predecessors have stopped using resources

In Section 3.3 we show that it is appropriate, for example, to combine different component removals into one aggregate activity and the corresponding shop repairs into another. Since some repairs can start before all of the removals have finished, the product transfer from the removal activity is more flow-like than event-based. Therefore the dependence relationships between aggregate activities cannot obey the "precedence requirement" criterion.

The "labor shop emphasis" criterion remains important since resource use must be modeled accurately. However, work location and technical system information at the key-op level may be aggregated. For example, we could make the following aggregations of key-ops:

- removal of condensate system and feed system components (two different systems, but related by resource use)
- painting of deck and hull (two locations with similar resource use).

Since an aggregate activity is a collection of many key-ops, it will operate at varying rates over time. Following Shephard [10], we represent the operation of an activity as a time history of *operating intensities.* Intensities are measured in some physical units; in our case they are units of resource use. We shall refer to the intensity time history of an aggregate activity as its *operating mode.*

A particular operating mode determines the time histories of resource use by the aggregate activity, and also time histories of intermediate product input and output. Being able to describe, and select from among, alternative operating modes is the basis for our approach to resource allocation and aggregate planning. In Section 3.2 we present activity operating modes in detail. Modeling product transfers in terms of restrictions on operating modes is discussed in Section 3.4.

3. DEVELOPMENT OF AGGREGATE OVERHAUL MODEL

The aggregate overhaul model consists of the following two components:

- a network of aggregate activities
- a representation of activity operation.

Its foundation is the activity analysis model of dynamic production networks, which is reviewed in the first section. The intensity history representation of activity operation is retained, but intermediate product transfers must be modeled differently in the aggregate activity network.

Section 3.2 discusses activity operation and demonstrates that key-op network aggregation has an effect on the accuracy of resource use measurement. This insight and other observations assist in the construction of an aggregate network, which is described in Section 3.3.

The special character of intermediate product transfer between activities is illustrated and its quantitative representation discussed in Section 3.4. Bounds on activity operation due to workspace limitations are dealt with in Section 3.5.

As we will see in Chapter 4, the aggregate overhaul model is part of a linear programming (*LP*) formulation that supports shipyard resource allocation. We consider computational issues of modeling activity operation in Section 3.6. The development of the aggregate model is concluded in Section 3.7 by presenting the version used in the *LP*.

3.1. Review of Dynamic Activity Analysis

The theory of dynamic production networks is due to Shephard [10]. We present a special case, *dynamic activity analysis,* as a starting point for development of the aggregate overhaul model. See Reference [6] for a more detailed treatment.

Intensity Function

A production system is viewed as a network of activities $A_1, A_2, \ldots, A_N$. The system operates during an interval $(0, T]$. We divide the interval into time periods $(t-1, t]$, $t=1,\ldots,T$, which are referred to simply by their index, t. The operation of each activity A_i is measured by a step function $z_i(t)$, $t=1,\ldots,T$, called the *intensity function* of A_i. The intensity at time t expresses the (non-negative) rate at which A_i operates during $(t-1, t]$ as measured in some physical unit.

Technical Coefficients

An activity uses resources $R_1, \ldots, R_K$ proportionally to its intensity function. It uses and produces intermediate products, also proportionally to $z_i(t)$. Let the intermediate products be denoted by IP_h, $h=1,\ldots,H$. A product IP_h may be produced or used by several activities, or may have a unique producer and user. The following technical coefficients of operation are given. They can vary over time, but we consider only the constant case here. Let

a_{ki} = amount of resource R_k used by A_i per unit intensity of A_i

c_{hi} = amount of intermediate product IP_h produced by A_i per unit intensity of A_i

$\bar{a}_{hi}$ = amount of intermediate product IP_h used by A_i per unit intensity of A_i.

Intensity Bounds

The operating intensity of activity A_i at time t is bounded by a parameter $\bar{z}_i(t)$ derived from known technical limitations on the operation of A_i. Thus,

$$z_i(t) \leqslant \bar{z}_i(t) \quad \text{for} \quad t=1,2,\ldots,T\ .$$

Intermediate Product Transfer Constraints

In the dynamic activity analysis model, intermediate products are produced proportionally to activity intensity, and "flow" to other activities through time. Production in one time period is assumed available for use in the next period. The transferred products may be stored until they are used. Transfers are constrained by requiring that intermediate product inventories are non-negative. In other words, the amount used of each product IP_h by time t may not exceed the amount produced by time $t-1$:

$$\sum_{j=1}^{N} \sum_{\tau=1}^{t} \bar{a}_{hj} z_j(\tau) \leqslant \sum_{i=1}^{N} \sum_{\tau=1}^{t-1} c_{hi} z_i(\tau) \qquad t=2,\ldots,T\ . \tag{3.1}$$

3.2. Characteristics of Aggregate Activity Operation

We represent an overhaul as a network of aggregate activities, and develop a modified dynamic activity analysis model of activity operation. In this section we assume that the aggregate network has been constructed, i.e., aggregate activities have been defined and each key-op has been assigned to such an activity. (Aggregation is discussed in Section 3.3.)

3.2.1. Activity Operating Modes

Let an aggregation of the key-ops $KO_1,...,KO_L$ to activities $A_1,...,A_N$ be given. We need to measure operating intensity in terms of some physical unit. The operation of an activity is strongly characterized by its use of labor resources. Since the resources (labor types) have the same physical unit we use the *total* resource application. Let

$$z_i(t) \equiv \frac{\textit{total amount of resources applied during } (t-1,t] \textit{ by activity } A_i}{\textit{total resource requirements of } A_i}$$

$$= \textit{proportion of total requirements that is used in } (t-1,t] \ .$$

The amount of labor resource R_k used by A_i is the sum of the amounts used by the key-ops assigned to it. For the above definition of intensity the coefficient a_{ki} (see Section 3.1) is given by $a_{ki} = \sum_l b_{kl}$, where the summation $\sum_l$ is always assumed to be over the key-ops assigned to A_i. Total resource use by A_i is denoted by $a_i \equiv \sum_k a_{ki} = \sum_l b_l$, where $b_l \equiv \sum_k b_{kl}$ is the total resource use by KO_l, and the summation $\sum_k$ is over all resources R_k, $k=1,...,K$.

We restrict the operation of A_i to those time periods in which the key-ops assigned to A_i can operate. This *window* of operation is denoted by $(S_i,F_i]$, where

$$S_i = \min_{KO_l \in A_i} ES_l \quad \text{and} \quad F_i = \max_{KO_l \in A_i} LF_l \ .$$

The operation of A_i within its window is represented by its *operating mode* $z_i(S_i), z_i(S_i+1),..., z_i(F_i)$. (For notational convenience we include $z_i(S_i)$, although $z_i(S_i)=0$ by construction.) We denote the operating mode by $OM_i \equiv \{z_i(t)\}_t$. Let

$$Z_i(t) \equiv \sum_{\tau=S_i}^{t} z_i(\tau) \tag{3.2}$$

be the cumulative operating intensity, i.e., the proportion of resources used up to time t. It satisfies $Z_i(S_i)=0$ and $Z_i(F_i)=1$. The operating mode of A_i can also be expressed in terms of the time history of cumulative intensity, $\{Z_i(t)\}_t$. (Throughout this paper we will interchangeably refer to $\{z_i(t)\}_t$ and $\{Z_i(t)\}_t$ as the operating mode of A_i.)

3.2.2. Modeling of Resource Use

Let $x_{ki}(t)$ be the amount of resource R_k applied to activity A_i during $(t-1, t]$, so that

$$x_{ki}(t) = a_{ki} z_i(t) \ .$$

Consider the activity resource load arising from a given schedule S of key-op start times. Let $x_{ki}^S(t)$ be the amount of R_k used by A_i at time t, and let $z_i^S(t)$ be the intensity of A_i at t, as derived from start schedule S. From (2.2) we can compute $x_{ki}^S(t)$ as

$$x_{ki}^S(t) = \sum_l x_{kl}(t) = \sum_{l \in \Lambda(S,t)} b_{kl} \bar{z}_l \ ,$$

where $\Lambda(S,t) \equiv \{l \mid KO_l \in A_i,\ S_l < t \leqslant F_l\}$ is the index set of all key-ops within A_i that operate at t, given S. The intensity, given S, is computed as

$$z_i^S(t) = \frac{\textit{total resource amount used by } A_i \textit{ during } (t-1,t],\ \textit{given } S}{\textit{total resource use by } A_i}$$

$$= \frac{\sum_k x_{ki}^S(t)}{a_i} = \sum_k \sum_{l \in \Lambda(S,t)} \frac{b_{kl}}{a_i} \bar{z}_l = \sum_{l \in \Lambda(S,t)} \frac{b_l}{a_i} \bar{z}_l \ . \tag{3.3}$$

We are interested in determining when resource use is modeled accurately in terms of intensities. Use of R_k by A_i during $(t-1, t]$, as derived from S, is $x_{ki}^S(t)$. Use of R_k, as modeled, is $a_{ki} z_i^S(t)$. Equating the two we have

$$x_{ki}^S(t) = a_{ki} z_i^S(t) \iff \sum_{l \in \Lambda(S,t)} b_{kl} \bar{z}_l = \sum_{l \in \Lambda(S,t)} a_{ki} \frac{b_l}{a_i} \bar{z}_l \ .$$

In general, equality will occur only if

$$b_{kl} = a_{ki} \frac{b_l}{a_i} \iff \frac{b_{kl}}{b_l} = \frac{a_{ki}}{a_i} \quad \text{for } k = 1, \ldots, K \ .$$

Consequently, overhaul resource use derived from a given schedule S is not modeled accurately if the *resource mixes*

$$\left\{ \frac{b_{kl}}{b_l} \right\}_{k=1,\ldots,K}$$

vary among key-ops KO_l assigned to the same activity.

3.2.3. Intensity Bounds

Consider the whole range of feasible key-op start time schedules between the early and late schedules *ES* and *LS*. The intensity of activity A_i at time t would be highest if all the key-ops assigned to A_i that *could* operate at t actually *did*. Let us define the intensity bound $\bar{z}_i(t)$ accordingly:

$$\bar{z}_i(t) \equiv \frac{\textit{total resource use during } (t-1, t] \textit{ by key-ops } KO_l \textit{ within } A_i \text{ for which } ES_l < t \leqslant LF_l}{\textit{total resource use by } A_i} .$$

The bound will overstate the highest possible intensity if some of the key-ops assigned to A_i are in series. In Section 3.5.1 we demonstrate that this bound is not very effective and discuss tighter intensity bounds.

3.2.4. Window Curves

The early and late key-op start schedules *ES* and *LS*, defined in Section 2.4, are the two extremes for overhaul execution as represented by the key-op network. We introduce an equivalent notion for the aggregate activity network. Let us define the *early operating mode* $OM_i^E = \{Z_i^E(t)\}_t$ of A_i by setting $S = ES$ in (3.3), as shown below:

$$Z_i^E(t) \equiv \sum_{\tau=S_i}^{t} z_i^E(\tau), \qquad z_i^E(\tau) \equiv \sum_{l \in \Lambda(ES,\tau)} \frac{b_l}{a_i} \bar{z}_l .$$

The *late operating mode* OM_i^L is defined analogously.

If the key-op aggregation were "perfect", i.e., for each A_i, the key-ops within A_i had the same resource mix, then the following would hold for any feasible operating mode of A_i:

- the proportion of total resources used by A_i up to time t could not exceed the proportion used in the early mode: $Z_i(t) \leqslant Z_i^E(t)$
- the proportion used by A_i after t could not exceed the proportion used in the late mode after t: $1-Z_i(t) \leqslant 1-Z_i^L(t)$.

Although the key-op aggregation is not likely to be "perfect", it is nevertheless reasonable to constrain OM_i by

$$Z_i^L(t) \leqslant Z_i(t) \leqslant Z_i^E(t) \quad \text{for} \quad t = S_i, \ldots, F_i . \tag{3.4}$$

(For $t=S_i$ and $t=F_i$ these constraints are trivially satisfied.)

Constraints of type (3.4) certainly constrain activity operation. However, further constraints are required; for example, we need to prevent an activity from operating early (in the model) although its predecessor operates late. Section 3.4 addresses this issue.

We refer to the early and late cumulative intensity curves as *window curves,* since all modes of activity operation must lie between them. Exhibit 3.1(a) shows typical early and late intensity curves, along with the intensity bound curve $\bar{z}_i(t)$. The cumulative intensity curves are shown in Exhibit 3.1(b).

At this point it is convenient to present an interpretation of the area between the window curves, which we denote by σ_i. It can be computed as

$$\sigma_i \equiv \sum_{t=S_i}^{F_i} Z_i^E(t) - Z_i^L(t) \ . \tag{3.5}$$

The magnitude of σ_i is clearly a measure of the latitude of operation of activity A_i. In fact, it can be interpreted as the *slack* of A_i, since it can be shown (by the definition of $Z_i^E(t)$ and $Z_i^L(t)$) that

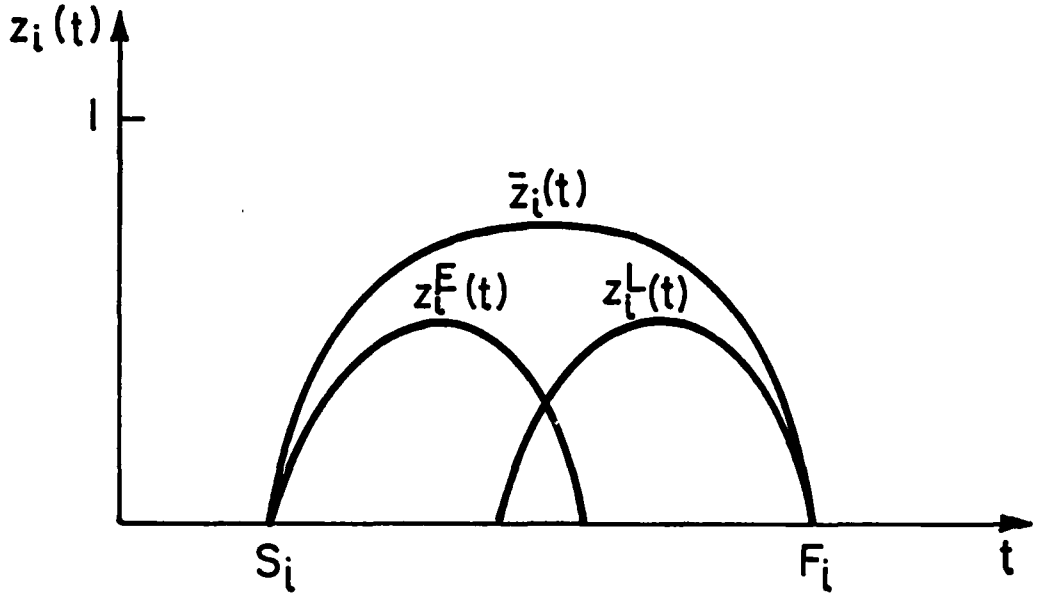
$$\sigma_i \equiv \sum_l \frac{b_l}{a_i} \sigma_l \ , \tag{3.6}$$

where σ_l is the slack of key-op KO_l (see Section 2.4.1). The area between the curves is thus the weighted average of the slacks of the key-ops within A_i.

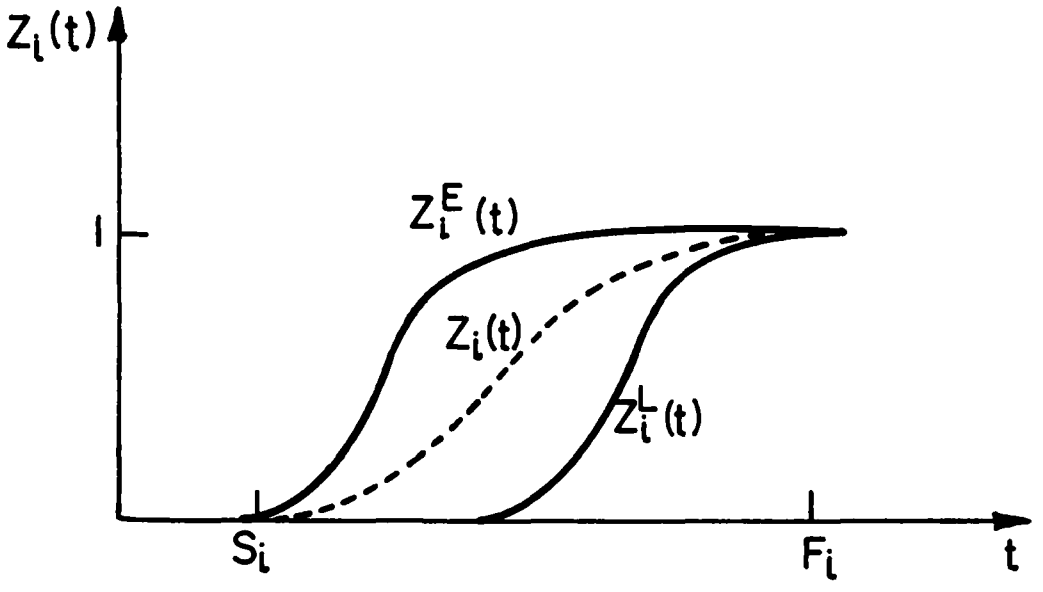
3.3. Aggregate Activity Network Construction

The aggregate activity network is a management-level representation of a ship overhaul. It identifies major overhaul components and the interaction between them. The network is an aggregation of the key-op network in the sense that each key-op is assigned to an aggregate activity. We seek to construct the network in a way that resource use can be modeled accurately.

We first review important observations about the aggregate network made in earlier sections. Then, in Section 3.3.2, guidelines for network construction are presented. In Section 3.3.3, we look at the types of intermediate product transfers that occur. Finally, we present in Section 3.3.4 an aggregate network constructed from actual ship overhaul data.



(a) Intensity Curves and Bound



(b) Cumulative Intensity Curves

EXHIBIT 3.1

EARLY AND LATE ACTIVITY OPERATION

3.3.1. Review of Earlier Observations

Resource Mix: In Section 3.2.2 we saw that the less the variation of resource mix among key-ops within an activity, the more accurately resource use of the aggregate activity will be represented.

Key Events: We identified key events as control points for transfer of intermediate products (Section 2.3). Important restraining events should be included in the aggregate network since they represent an event-based transfer of product between activities. Some monitoring events will be implicit in the network when they refer to the completion of intermediate product transfer between two aggregate activities.

Work Breakdown Criteria: In Section 2.5 we saw that the level of detail which characterized the key-op network must be reduced. In particular, we cannot distinguish all different technical systems and work locations. While the strict precedence relationships between key-ops are not captured in the aggregate model, we do need to extract as much information as possible about the operating alternatives of key-ops. Clearly, resource use characteristics remain important as a criterion for activity definition.

3.3.2. Guidelines for Network Construction

To construct the aggregate activity network we must analyze the structure of the key-op network. This will allow us to

- define aggregate activities (network nodes)
- identify intermediate product transfers between activities (network arcs).

This analysis is not easily systematized. It certainly does not proceed in a sequential order of steps. Rather, it is an iterative process by which the network is continually refined.

Exhibit 3.2 shows the key-op network for a simple overhaul. (The lines around pairs of nodes should be ignored initially.) Although it is unrealistically straightforward (and made to measure) it does help clarify the steps presented below. The example contains four technical systems. Two are mechanical ($M1, M2$), the other two electrical ($E1, E2$). Each system overhaul involves inspection (I), repair (R), test (T), and work space clean-up (C). All inspections must be completed before any repairs can be started. Clean-up and test can both start after repair is finished, and they are assumed not to interfere with each other. Repair and test of each system involve the same labor type, but in different quantities and rates. All clean-ups require the same labor type.

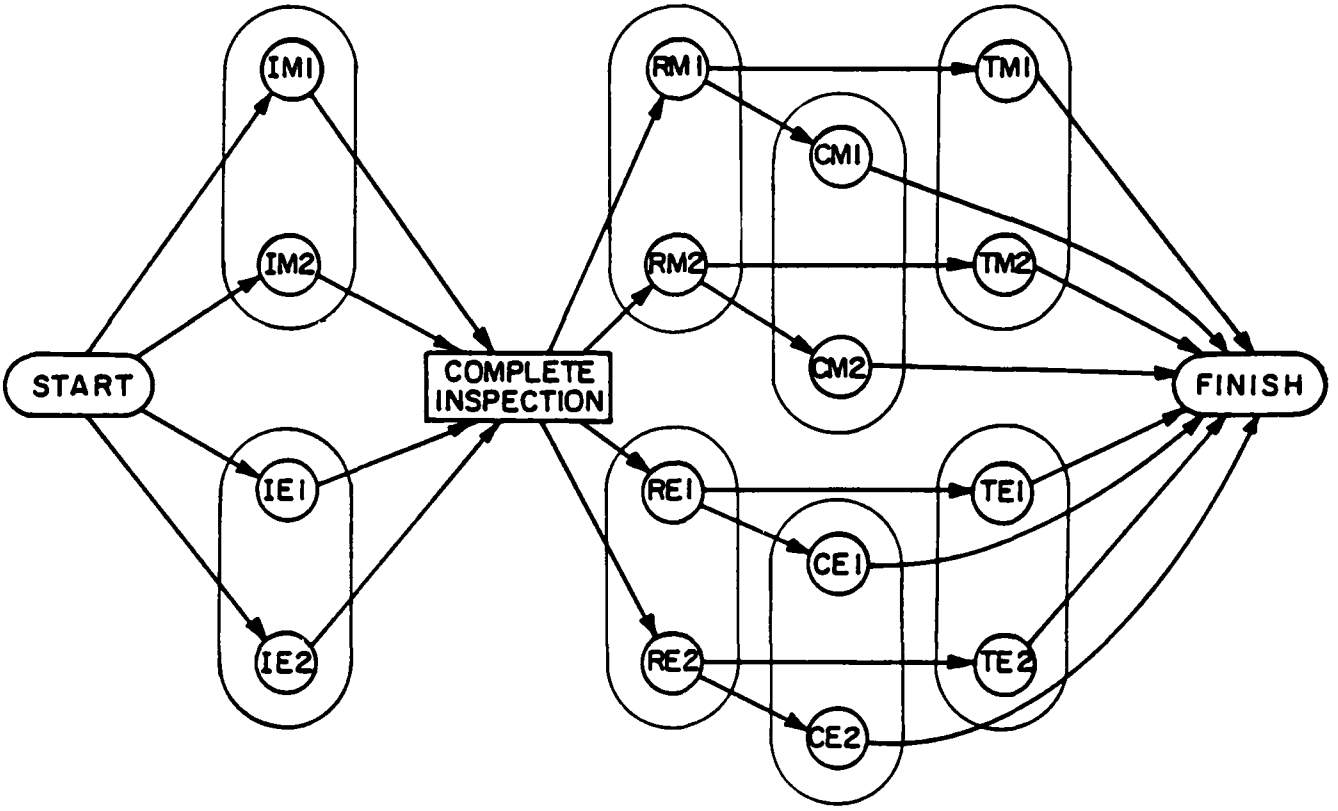


EXHIBIT 3.2

EXAMPLE - KEY-OP NETWORK

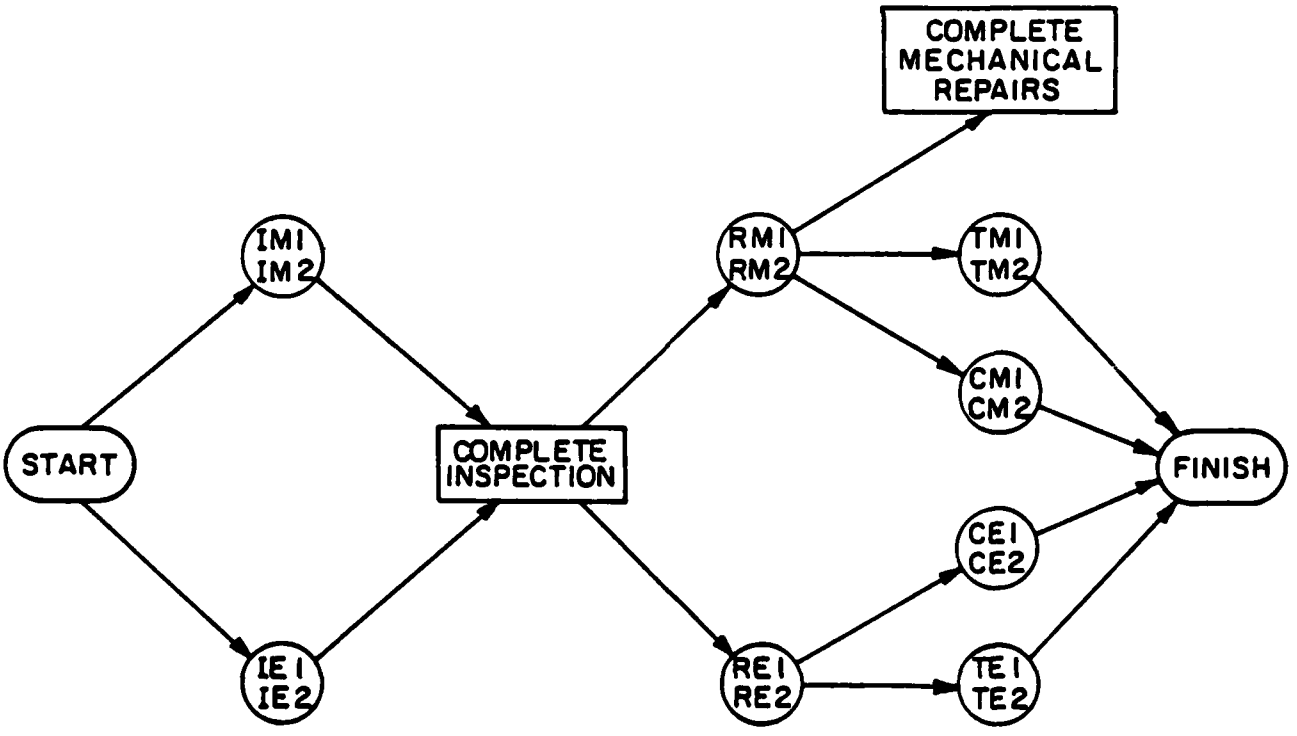


EXHIBIT 3.3

EXAMPLE - AGGREGATE ACTIVITY NETWORK

The following steps describe the process of developing the aggregate network. As mentioned above, some iteration of these steps is required.

Step 1: Identify major restraining key events and how they restrict the overhaul of each technical system.

Example: As mentioned, the "Complete Inspection" event succeeds all inspections, and precedes all repairs.

Step 2: For subnetworks between key events, group key-ops within similar technical systems:

Example: We identify four groups:

- inspection of mechanical systems ($IM1,2$)
- inspection of electrical systems ($IE1,2$)
- overhaul of mechanical systems ($RM1,2$; $CM1,2$; $TM1,2$)
- overhaul of electrical systems ($RE1,2$; $CE1,2$; $TE1,2$).

Step 3: Within each group, distinguish between different labor shop emphases. Identify differences in timing and resource use rate among the tasks with similar labor shop emphasis.

Example: Exhibit 3.2 shows the resulting groups of tasks. Note that repair and test are distinct since they use the same labor type but at different times and at different rates.

Step 4: Connect groups to show intermediate product transfers. (The next section discusses this in more detail.)

Example: Network arcs between the groups in Exhibit 3.2 are easy to identify. Exhibit 3.3 shows the resulting aggregate activity network.

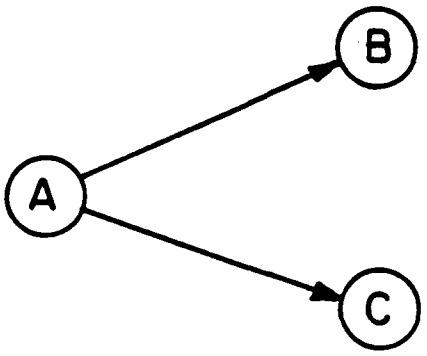
Step 5: Identify monitoring key events and connect activities with them.

Example: Suppose mechanical repairs are to be monitored closely. Add an event node and connect the activity with it, as in Exhibit 3.3.

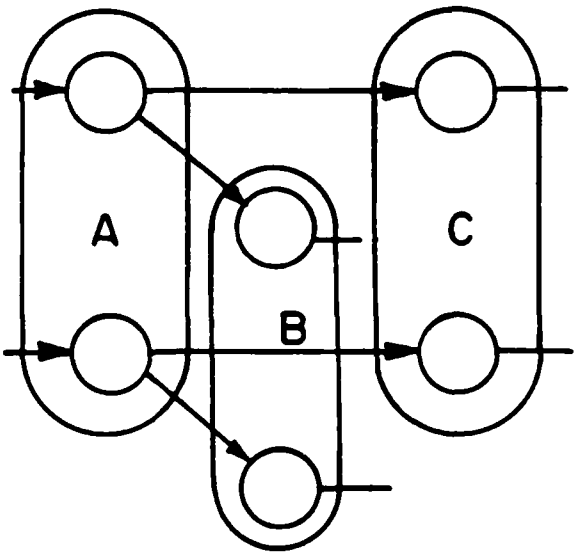
3.3.3. Intermediate Product Transfer Arcs

Analysis of the intermediate product transfers within a preliminary aggregate network (developed according to the guidelines in Section 3.3.2) may lead to revisions in the aggregation, or further aggregation. The modeling of product transfers for various types of key-op aggregation is discussed below.

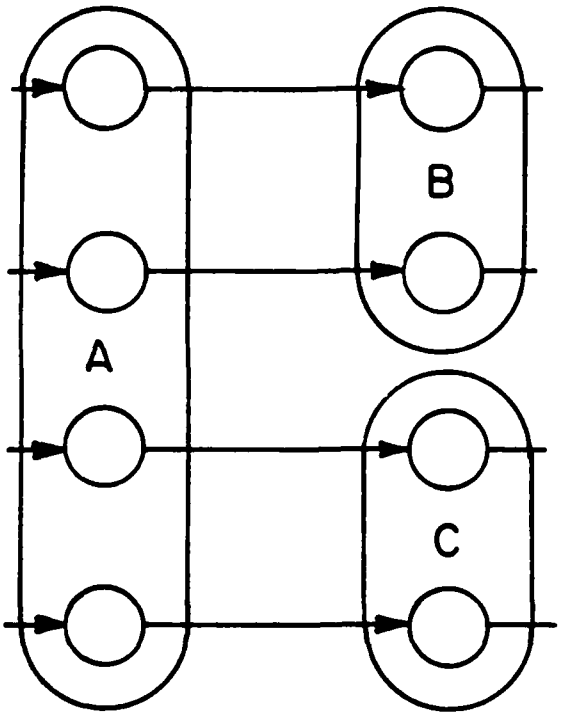
An arc connects two aggregate activities A and B if a key-op in A has a successor in B, or equivalently, if intermediate product is transferred from a key-op in A to a key-op in B. As



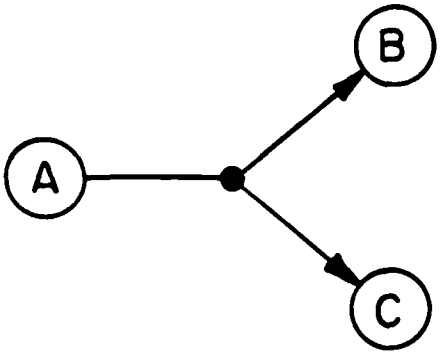
(a) Aggregate Network
-Distinct Outputs



(b) Key-op Network
-Distinct Outputs



(c) Key-op Network
-Shared Output



(d) Aggregate Network
-Shared Output

EXHIBIT 3.4

DISTINCT AND SHARED INTERMEDIATE PRODUCT OUTPUT

stated in Section 2.2, a key-op produces a distinct product for each of its successor key-ops.

A particular key-op aggregation may result in transfers from key-ops in A to key-ops within *many* other activities. We could ignore unimportant transfers from activity A. Such a situation may indicate, however, that the aggregation could be improved. The aggregate activities represent major overhaul components, so that only a few transfer arcs from each activity to other activities should be necessary.

Activity A produces intermediate products for its successors. As is discussed below, it may produce one product which is shared by its successors, or a distinct product for each successor. A given operating mode of A implies a distribution of intermediate product output through time. By the nature of the operating mode model, this distribution is identical for all outputs that A produces.

For convenience of exposition, let activity A have two successors B and C, as in Exhibit 3.4(a). (The analysis that follows can be extended to more than two successors.) We study the product output of A by considering the key-op subnetwork underlying the activities A, B, and C.

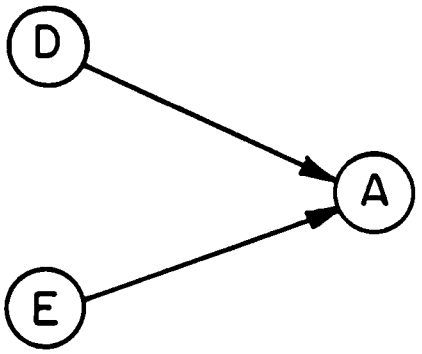
Distinct Products

Suppose A, B, and C are aggregated from the key-op subnetwork in Exhibit 3.4(b). Each of the key-ops in A produces distinct products for its successors in B and C. Accordingly, the intermediate products for B and C are considered to be distinct.

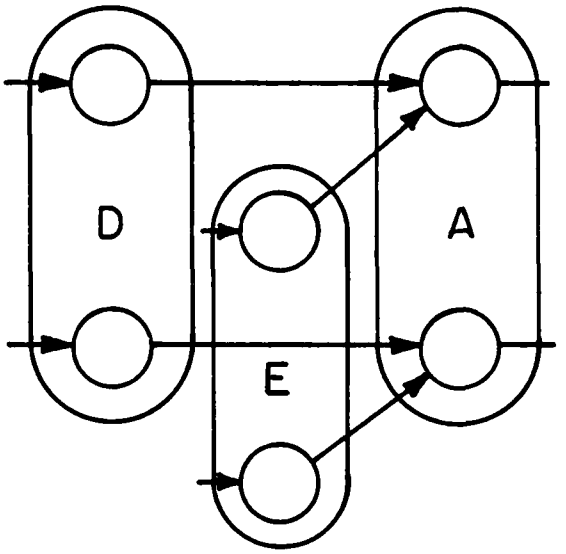
Shared Intermediate Product Outputs

Now suppose they are aggregated as in Exhibit 3.4(c). Let us look at an example. Suppose the key-ops in A are removals of four components within the same technical system. Two of the components must then be painted (B), the other two lubricated (C). We can plan the component removals in A, i.e., its use of resources over time, independently of the *order* of removals. Thus, for resource use planning, the removals can be considered as indistinguishable, and also their output.

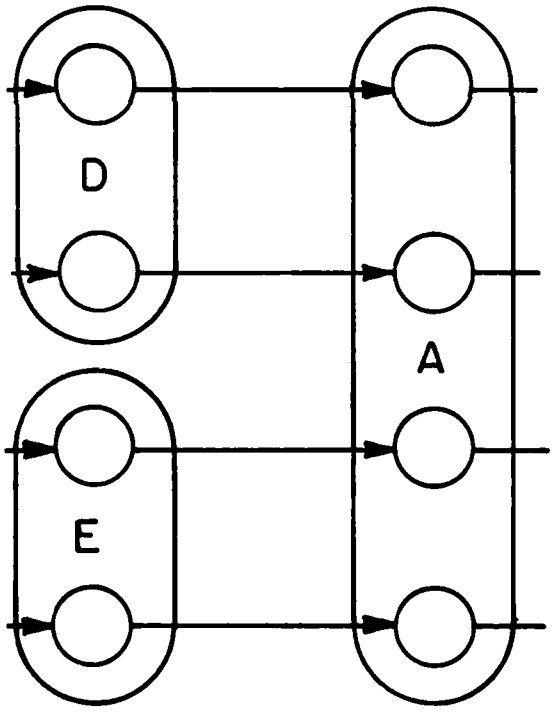
We therefore assume that only one intermediate product is produced by A, which is then *shared* by B and C. This is indicated graphically in Exhibit 3.4(d). As will be discussed in Section 3.4, the operation of B and C must be constrained so as to properly share the product output of A.



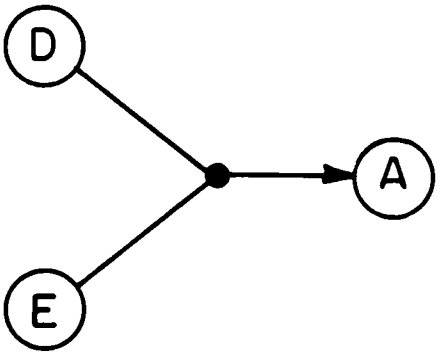
(a) Aggregate Network
-Distinct Inputs



(b) Key-op Network
-Distinct Inputs



(c) Key-op Network
-Pooled Inputs



(d) Aggregate Network
-Pooled Inputs

EXHIBIT 3.5

DISTINCT AND POOLED INTERMEDIATE PRODUCT INPUT

Pooled Intermediate Product Outputs

Analogous arguments can be made if activity A has multiple predecessors. A given operating mode of A implies a particular distribution of intermediate product input. We assume that this distribution is identical for all distinct inputs that A uses. Suppose it has two predecessors D and E, as in Exhibit 3.5(a). The intermediate products from D and E are considered distinct if the underlying key-op subnetwork is similar in structure to Exhibit 3.5(b). We say that D and E *pool* their outputs if the key-op subnetwork is as in Exhibit 3.5(c). Activity A then receives the pooled output as a single input, as indicated in Exhibit 3.5(d).

Consider the aggregate network in Exhibit 3.3. The clean-up activities are, by assumption, similar in content. The intermediate product that each clean-up activity receives is a workspace that is to be put back in order. In such a case we can merge the two clean-up activities. The product input consists of a *pool* of workspaces that are "produced" by the repair activities. Exhibit 3.6 shows the aggregate network after the activity merger. (This portion of the example aggregation was not included in Section 3.3.2 to avoid confusion.)

3.3.4. Aggregate Network from Overhaul Data

An aggregate activity network for an actual ship overhaul was constructed. The original key-op network contained approximately 1150 key-ops. From this network a more aggregate network was developed with approximately 430 key-ops. These 430 more aggregate key-ops were in turn aggregated into 39 activities. A list of restraining events is given in Exhibit 3.7, and the 12 largest labor shops are listed in Exhibit 3.8. Exhibits 3.9 and 3.10 show the aggregate network and activity data, consisting of

- the number of key-ops assigned to activity A_i
- its "lead shop" (see Section 2.2)
- its total labor requirement, a_i (in man-days)
- the range of its operating window, $(S_i, F_i]$ (in two week units).

Three main groups of technical systems were identified, namely Mechanical, Structural, and Electrical & Electronic. Within each group, differences in timing and labor use rate had to be taken into account, for example:

- some structural overhauls could take place at any time between docking and undocking (activity A_8), so they could not be aggregated together with the removal, repair, and reinstallation sequence (A_2, A_3, A_4) for the structural component overhauls

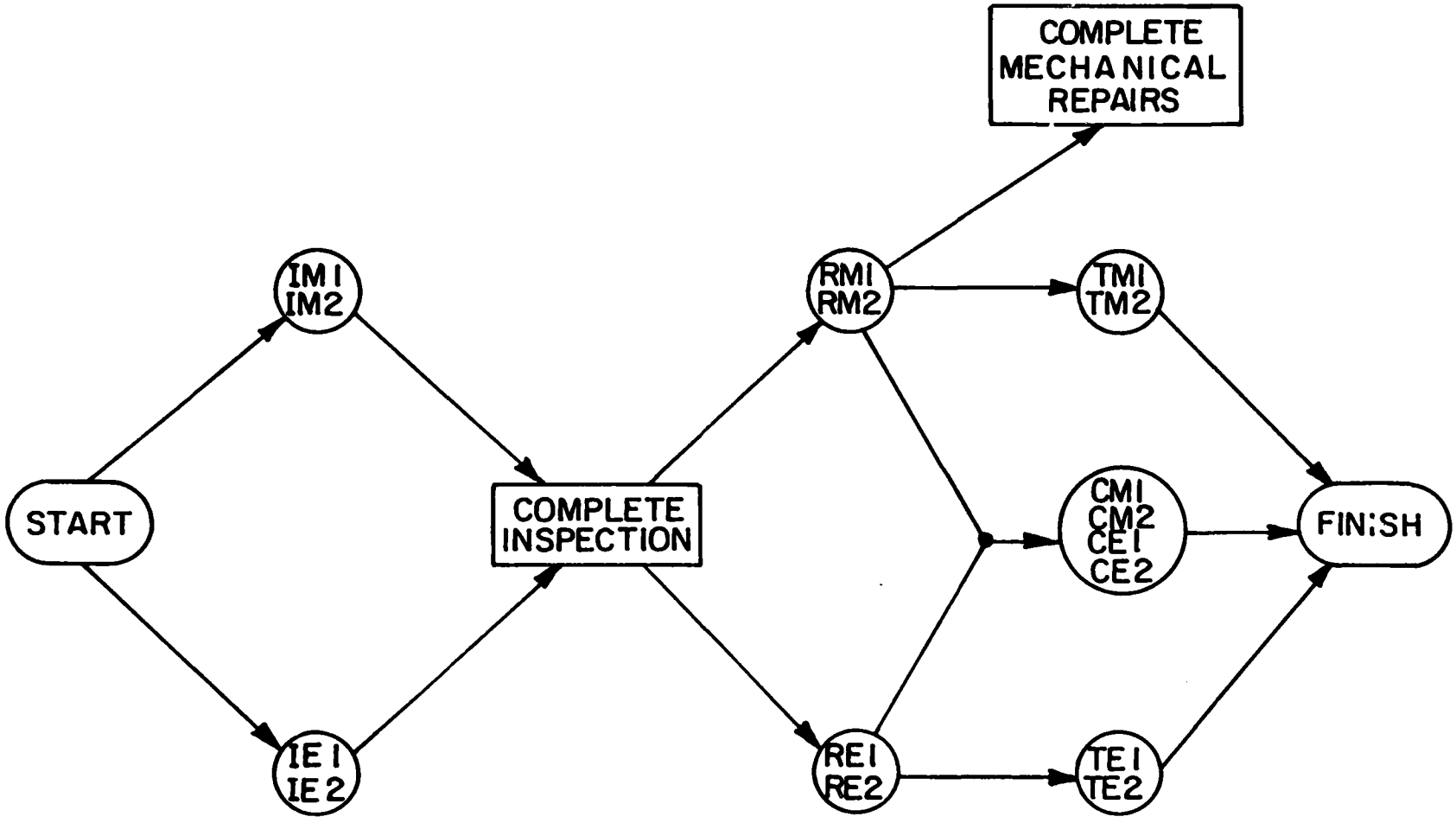


EXHIBIT 3.6

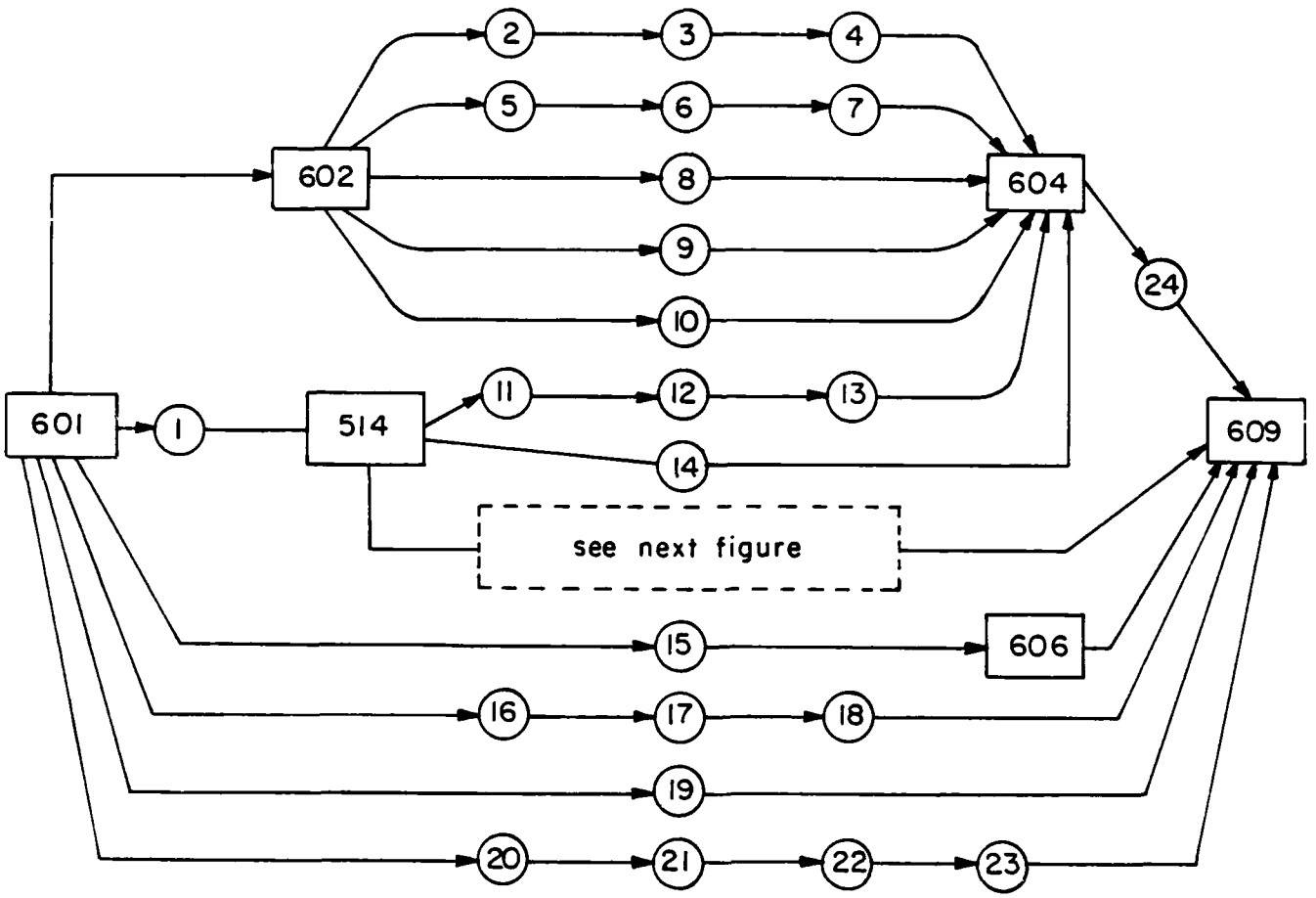
EXAMPLE - REVISED AGGREGATE ACTIVITY NETWORK

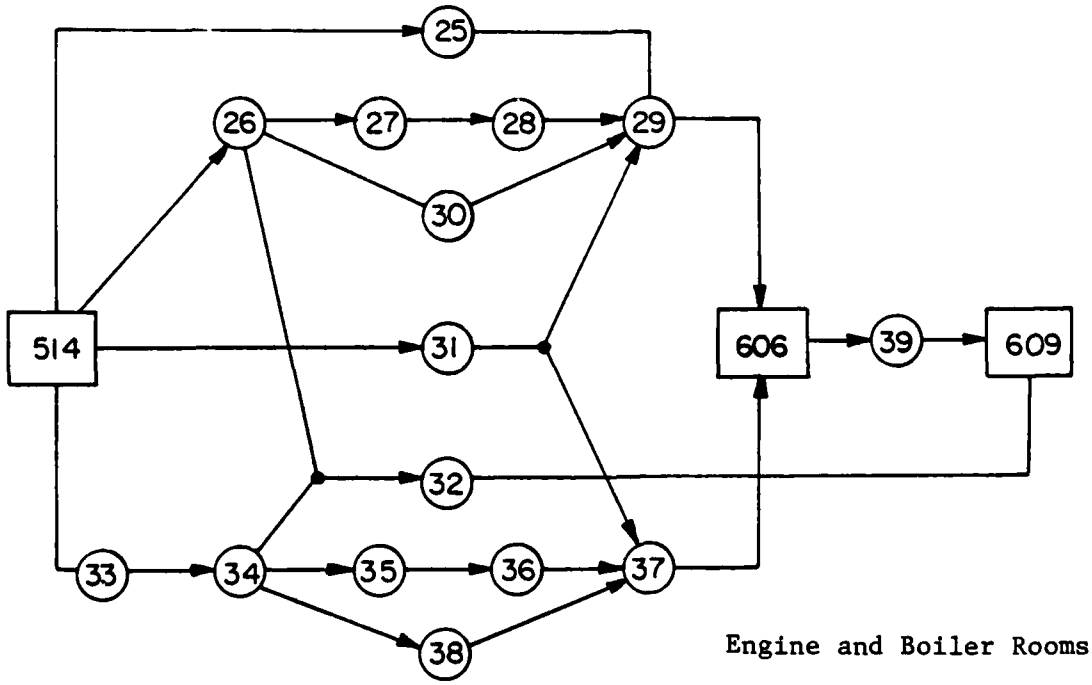
EXHIBIT 3.7 OVERHAUL DATA - LIST OF RESTRAINING EVENTS

Code	Key Event	Occurrence Time (two week units)
601	Start Overhaul	0
514	Complete Lagging Removal	2
602	Dock Ship	5
604	Undock Ship	14
606	Prepare Engine & Boiler Rooms for Test	16
609	Complete Production Work	20

EXHIBIT 3.8 OVERHAUL DATA - LIST OF LABOR SHOPS

No.	Shop Name	Total Req'ts (man-days)
1	Pipefitting	5913
2	Mechanical Group - Shipboard	5111
3	Mechanical Group - Shore	4307
4	Electrical	3038
5	Painting	2999
6	Electronics	2796
7	Boilermakers	2309
8	Weld & Burn	2289
9	Structural Group I	1392
10	Structural Group II	1333
11	Shipwright	638
12	Rigging	536
	TOTAL	32660





Engine and Boiler Rooms

EXHIBIT 3.9

OVERHAUL DATA - AGGREGATE ACTIVITY NETWORK

EXHIBIT 3.10 OVERHAUL DATA - LIST OF ACTIVITIES

i	Activity A_i	No. of Key-ops	Lead Shop	a_i	S_i	F_i
1	Remove Lagging, Engine & Boiler Rooms	5	1	500	0	2
2	Remove Structural Components 602-604	11	2	105	5	9
3	Repair Structural Components 602-604	18	5	508	6	13
4	Reinstall Structural Components 602-604	13	8	152	10	14
5	Remove Mechanical Components 602-604	8	2	127	5	9
6	Repair Mechanical Components 602-604	13	2	505	7	12
7	Reinstall and Test Mechanical Components 602-604	10	2	217	9	14
8	Misc. Mechanical Overhauls 602-604	18	2	548	5	14
9	Misc. Electrical Overhauls 602-604	32	4	436	5	14
10	Misc. Structural Overhauls 602-604	18	8	660	5	14
11	Misc. Removals 514-604	10	2	52	1	6
12	Misc. Repairs 514-604	17	3	495	2	13
13	Misc. Reinstallations & Test 514-604	19	2	197	6	14
14	Overhaul Tanks and Voids	40	5	1668	1	14
15	Overhaul Air Systems	44	1	885	0	16
16	Misc. Removals 601-609	29	2	205	0	12
17	Misc. Repairs 601-609	33	2	764	5	13
18	Misc. Reinstallations 601-609	29	2	696	8	19
19	Misc. Component Overhauls 601-609	53	1	1917	0	19
20	Remove Electrical System Components	45	4	331	0	9
21	Repair Electrical System Components	84	6	2597	1	15
22	Reinstall Electrical System Components	61	4	1046	3	16
23	Test Electrical System Components	45	6	1104	5	20
24	Post Undocking Tests	23	5	190	14	19
25	In-place Overhaul Engine Room Components	51	2	1876	1	15
26	Remove Engine Room Components	68	2	575	1	10
27	Repair Engine Room Components	92	3	3207	2	14
28	Reinstall Engine Room Components	70	2	1193	3	15
29	Test Engine Room Components	44	2	1064	7	16
30	In-place Repair Engine Room Components	14	2	387	3	15
31	Overhaul Vents & Bilges	11	9	1385	1	14
32	Repair & Reinstall Lagging	7	1	1577	5	19
33	Inspect Boiler	8	7	467	1	4
34	Remove Boiler Room Components	11	7	624	2	9
35	Repair Boiler Room Components	15	7	813	3	12
36	Reinstall Boiler Room Components	16	7	1161	5	15
37	Test Boiler Room Components	21	7	733	9	16
38	In-place Repair Boiler Room Components	32	7	1579	3	16
39	System Test Turbine & Boiler	9	2	114	16	19
	TOTALS	1147		32660		

- *two* engine room in-place repair activities were defined (A_{25} and A_{30}) since some in-place repairs succeeded component removals, but others did not.

Intermediate products have unique producers and users, except for two cases:

- the bilge and ventilation overhauls in the engine and boiler rooms were combined since they are indistinguishable (A_{31}); this activity shares its output with the test activities in engine and boiler rooms (A_{29}, A_{37})
- lagging (asbestos insulation) removal in the engine and boiler rooms has to be completed before component removals can begin, as key event 514 indicates. Once the component removals (A_{26}, A_{34}) have been completed, the lagging can be repaired and eventually reinstalled (A_{32}). The removal activities A_{26} and A_{34} pool their output for A_{32}.

Note that, for example, the engine room removal activity (A_{26}) produces distinct intermediate products for its successors A_{27} and A_{30}. While A_{27} is the shop repair of removed components, A_{30} is the in-place repair of the system components not removed. The underlying key-op network resembles that in Exhibit 3.4(b).

Monitoring events have not been included in Exhibit 3.9. They correspond to the completion of one or more activities. For example, shop repairs of engine and boiler room components are monitored together, so that A_{27} and A_{35} both would lead to a single monitoring event node.

The activity window curves were computed for given key event dates. Exhibit 3.7 shows the restraining event dates, and Exhibit 3.10 the activity window ranges. They are all in units of ten working days. (The reason for this becomes clear in Section 3.7.)

The aggregate network, as it was constructed, shows too much detail in some places, perhaps, and not enough in others. To some extent this is due to the authors' lack of familiarity with the actual overhaul. On the other hand, the aggregation is only as good as the underlying key-op network. In fact, the current method of developing key-op networks does not adequately take labor use into account. Research is underway to improve the key-op network development methodology.

3.4. Representation of Intermediate Product Transfer

We are interested in describing the feasible distributions of resource use by an overhaul. Each distribution represents a particular set of operating modes of the activities within the overhaul. In Section 3.2 we developed constraints on the operation of the activities by considering each activity separately. Unless we constrain the operation of all activities to be consistent with one another, our model of overhaul resource use will be inadequate and inaccurate.

The operation of the activities will be consistent if intermediate product transfers are feasible, i.e., intermediate products are available when they are required. At the key-op level this simply means that key-ops must finish before their successors start. However, at the aggregate activity level it is not possible to represent product transfers in such detail; rather, we must model the transfers by constraining the set of operating modes appropriately. Developing such constraints is the topic of this section. We begin by presenting a general model of intermediate product transfers.

3.4.1. General Model of Intermediate Product Transfers

The operating mode $OM_i=\{Z_i(t)\}_t$ of an aggregate activity A_i determines its distribution of resource use. In addition, OM_i implies a particular distribution of the intermediate products used and produced by A_i. The activity may be the sole user or producer of an intermediate product, or one of several users or producers.

We denote the intermediate products by $IP_1, \ldots, IP_H$. As in Section 3.1, let

$\bar{a}_{hi}$ = *amount of* IP_h *that* A_i *uses*

c_{hi} = *amount of* IP_h *that* A_i *produces* .

Since the total amounts used and produced of each intermediate product are equal, we can assume the coefficients to be normalized as follows:

$$\sum_{i=1}^{N} \bar{a}_{hi} = \sum_{i=1}^{N} c_{hi} = 1 \qquad h=1,\ldots,H \ .$$

As mentioned in Section 3.3.3, we also assume that the distribution of intermediate product use by A_i is identical for all IP_h for which $\bar{a}_{hi}>0$. Let us denote their common *input distribution* by $\{I_i(t)\}_t$, where for $t=1, \ldots, T$,

$I_i(t)$ = proportion of $\bar{a}_{hi}$ used by A_i up to time t, given OM_i (applying to all IP_h, $h=1,...,H$).

Similarly, let $\{O_i(t)\}_t$ be the common *output distribution* of intermediate products produced by A_i, where for $t=1, \ldots, T$,

$O_i(t)$ = proportion of c_{hi} used by A_i up to time t, given OM_i (applying to all IP_h, $h=1,...,H$).

Outside of the operation window, the following holds:

$$I_i(t) = O_i(t) = 0 \qquad t \leqslant S_i$$
$$I_i(t) = O_i(t) = 1 \qquad t \geqslant F_i.$$

The input and output distributions of A_i are functions of the operating mode of A_i. In the dynamic activity analysis model, for example,

$$I_i(t) = Z_i(t) \ , \ O_i(t) = Z_i(t-1) \ .$$

The operating modes of all activities are consistent with one another if, for each intermediate product IP_h and each time period t, total use of IP_h up to t (as induced by the operating modes) does not exceed the induced total production of IP_h up to t. This is expressed as follows:

$$\sum_{j=1}^{N} \bar{a}_{hj} I_j(t) \leqslant \sum_{i=1}^{N} c_{hi} O_i(t) \qquad h=1,...,H; \ t=1, \ldots, T \ . \tag{3.7}$$

Note that this reduces to (3.1) for the dynamic activity analysis model. In the aggregate activity network the following three cases of the above constraints can occur, depending on the number of users and producers of each intermediate product:

(1) *Single Producer and User*

IP_h is produced only by A_i ($c_{hi}=1$) and used only by A_j ($\bar{a}_{hj}=1$):

$$I_j(t) \leqslant O_i(t) \ . \tag{3.8a}$$

(2) *Single User, Pooled Production*

IP_h is used only by A_j ($\bar{a}_{hj}=1$) but is produced by multiple predecessors of A_j (those activities A_i for which $c_{hi}>0$):

$$I_j(t) \leqslant \sum_{i=1}^{N} c_{hi} O_i(t) \ . \tag{3.8b}$$

(3) *Single Producer, Shared Use*

IP_h is produced only by A_i (c_{hi}=1) but is shared by multiple successors of A_i (those activities A_j for which $\bar{a}_{hj}>0$):

$$\sum_{j=1}^{N} \bar{a}_{hj} I_j(t) \leqslant O_i(t) \ . \tag{3.8c}$$

Ideally, we would like to find explicit representations of the relationship between use and production of intermediate product and the operating mode of each aggregate activity. Constraints (3.8) could then be stated directly in terms of operating intensities. In Section 3.4.2 we study this relationship, but conclude that the correspondence between resource use and intermediate product input and output cannot be easily described in the aggregate overhaul model. As an alternative, we present constraints in Section 3.4.4 that describe the set of consistent operating modes by comparing the "relative progress" of the activities. The resulting relative progress transfer constraints turn out to be analogous to those in (3.8).

3.4.2. Product Transfer Characteristics

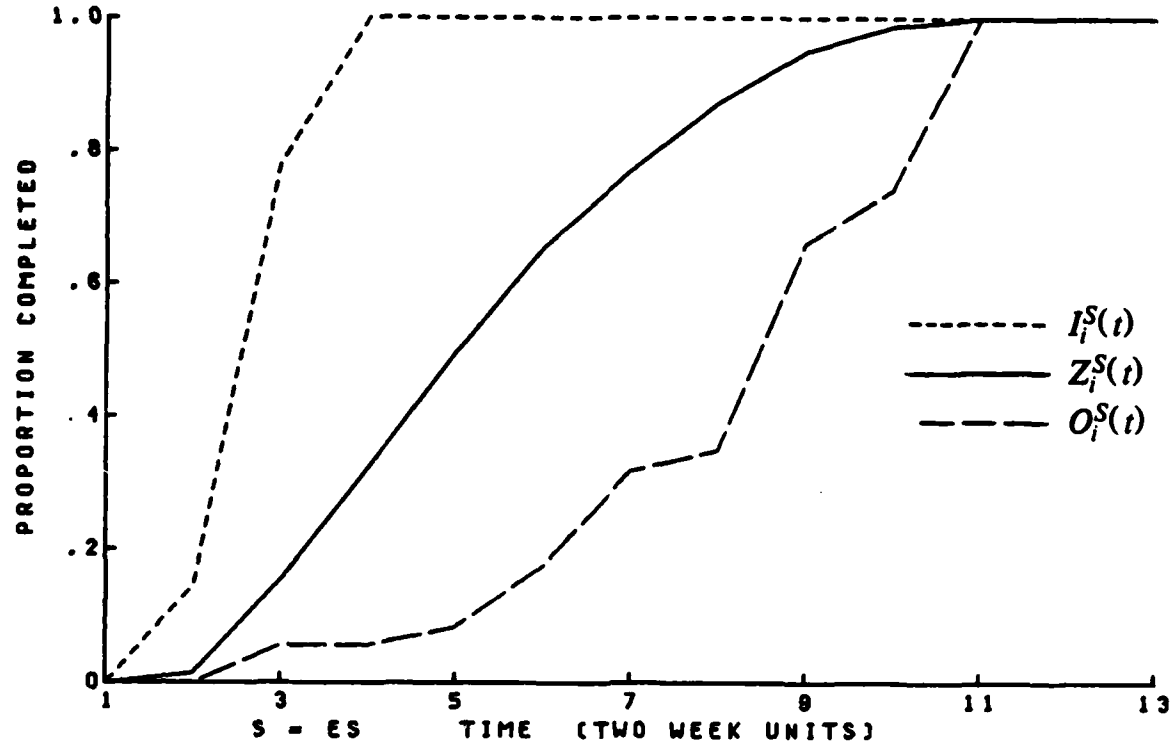
Given a key-op start time schedule $S=\{S_1,...,S_L\}$, we approximate the distributions $\{I_i^S(t)\}_t$ and $\{O_i^S(t)\}_t$ induced by S in terms of resource quantities. They will be compared with $\{Z_i^S(t)\}_t$ using examples from actual overhaul data.

A key-op KO_l within A_i uses its intermediate product input when it starts. Let us measure the input quantity by the amount of resources that is *committed* when KO_l starts, namely b_l. We approximate the distribution of intermediate product use by the distribution of resources committed by A_i. The proportion of intermediate products used is thus taken as the proportion of resources committed:

$$I_i^S(t) = \sum_{l \in \Gamma(S,t)} \frac{b_l}{a_i}, \quad \textit{where}\ \Gamma(S,t) \equiv \{l \mid KO_l \in A_i,\ S_l < t\} \ .$$

Recall that the input distributions of all intermediate products used by A_i are assumed to be identical (and equal to $\{I_i^S(t)\}_t$). To avoid complications due to multiple product inputs we will compare $\{I_i^S(t)\}_t$ with $\{Z_i^S(t)\}_t$ for an activity from the overhaul data which has only one predecessor, i.e., one intermediate product input.

Analogously, a key-op KO_l within A_i produces its intermediate product output when it finishes, and we measure the output quantity by the amount of resources that is *released* when



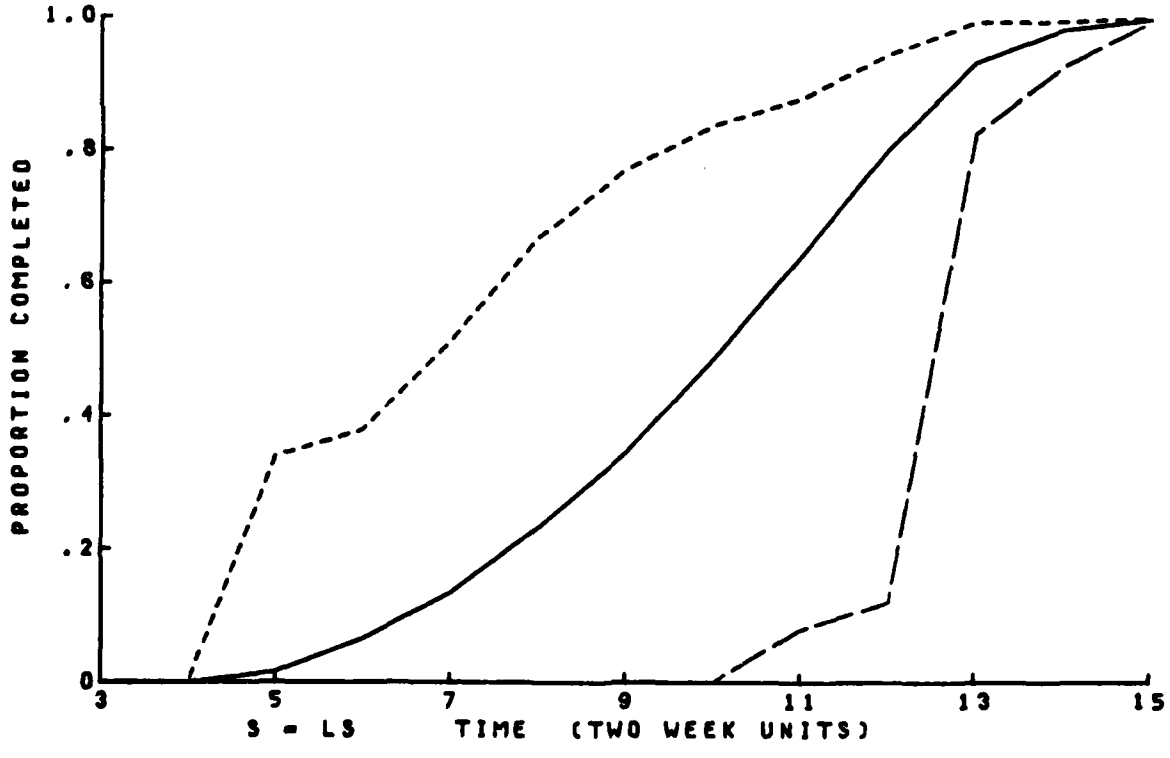


EXHIBIT 3.11

RESOURCE USE VS. INTERMEDIATE PRODUCT INPUT AND OUTPUT

KO_l finishes, namely b_l. The proportion of intermediate product produced is approximated by the proportion of resources released:

$$O_i^S(t) = \sum_{l \in \Delta(S,t)} \frac{b_l}{a_i}, \quad \text{where } \Delta(S,t) \equiv \{l \mid KO_l \in A_i,\ S_l + D_l \leqslant t\} .$$

As before, we will compare $\{O_i^S(t)\}_t$ with $\{Z_i^S(t)\}_t$ for a given schedule S, choosing an activity with only one successor.

Exhibit 3.11 shows $\{I_i^S(t)\}_t$, $\{Z_i^S(t)\}_t$, and $\{O_i^S(t)\}_t$ for an activity from the overhaul data (activity A_{27} in Exhibit 3.9) which does have only one predecessor and one successor. In Exhibit 3.11(a), S is chosen to be the early schedule ES; in Exhibit 3.11(b), $S=LS$. We observe that intermediate product input and output are certainly not proportional to resource use, as the dynamic activity analysis model assumes. Also, there is no constant time lag or resource use lag between $\{I_i^S(t)\}_t$ and $\{Z_i^S(t)\}_t$, or between $\{Z_i^S(t)\}_t$ and $\{O_i^S(t)\}_t$.

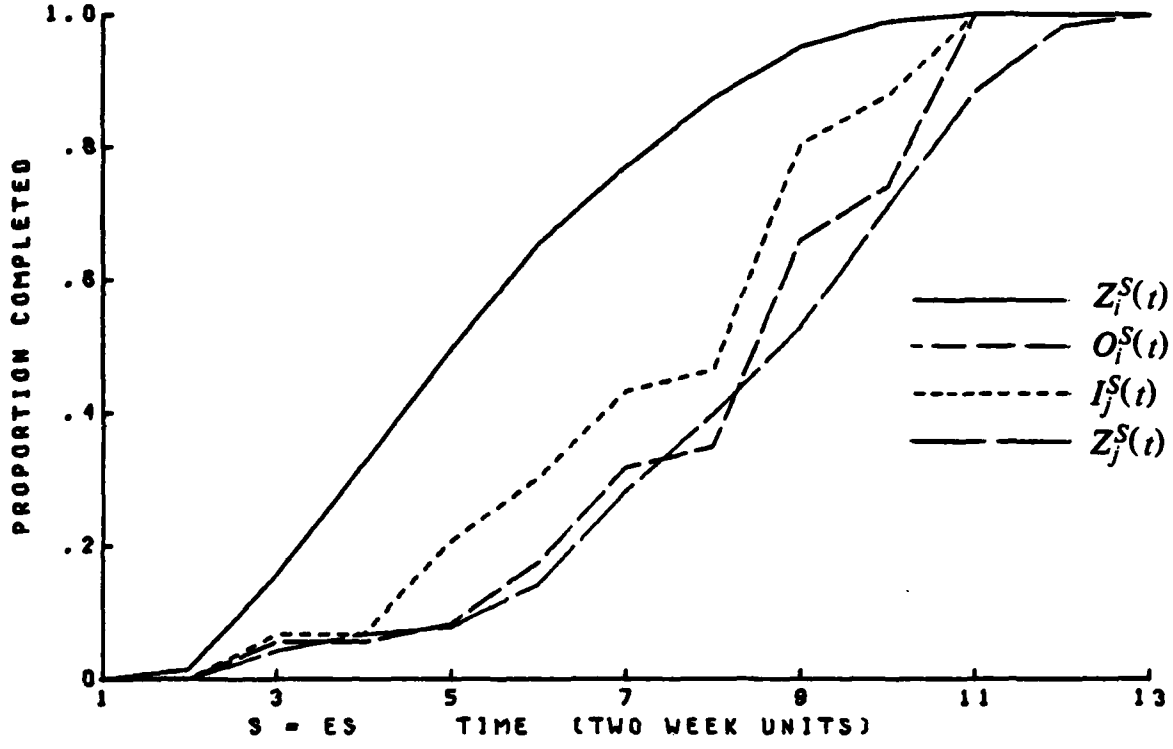
To illustrate the transfer of intermediate product we introduce the following notion: an activity A_j is said to *operate tightly* with its predecessors (i.e., suppliers of intermediate product) if it uses its intermediate product inputs as soon as they are produced. At the key-op level, A_j operates tightly with its predecessors if all key-ops within A_j start immediately upon completion of their predecessor key-ops.

If A_j has a predecessor A_i, but no other activities precede A_j or succeed A_i, then the following should hold when A_j operates tightly with A_i:

$$I_j(t) = O_i(t) \qquad t = S_i, \ldots, F_i .$$

Referring to the overhaul data again, the activity just discussed was chosen so that its successor has no other predecessors. Exhibit 3.12 shows $\{O_i^S(t)\}_t$ and $\{I_j^S(t)\}_t$ for $S=ES$ and $S=LS$. (Note that for the early and late schedules, all activities operate tightly with their predecessors.) We observe that the curves are quite close. Differences inevitably occur when, for example, KO_m in A_j succeeds KO_l in A_i, but their respective contributions b_l/a_i and b_m/a_j to resource use of A_i and A_j are unequal.

Exhibit 3.12 also shows the operating modes $\{Z_i^S(t)\}_t$ and $\{Z_j^S(t)\}_t$ for $S=ES$, LS. There is no constant time lag or resource use lag between the two. Furthermore, while all transfers take place in the overlap $(S_j, F_i]$ of the windows of A_i and A_j, the distribution of product transfer within the overlap can vary strongly. Note that the transfers determine the operation



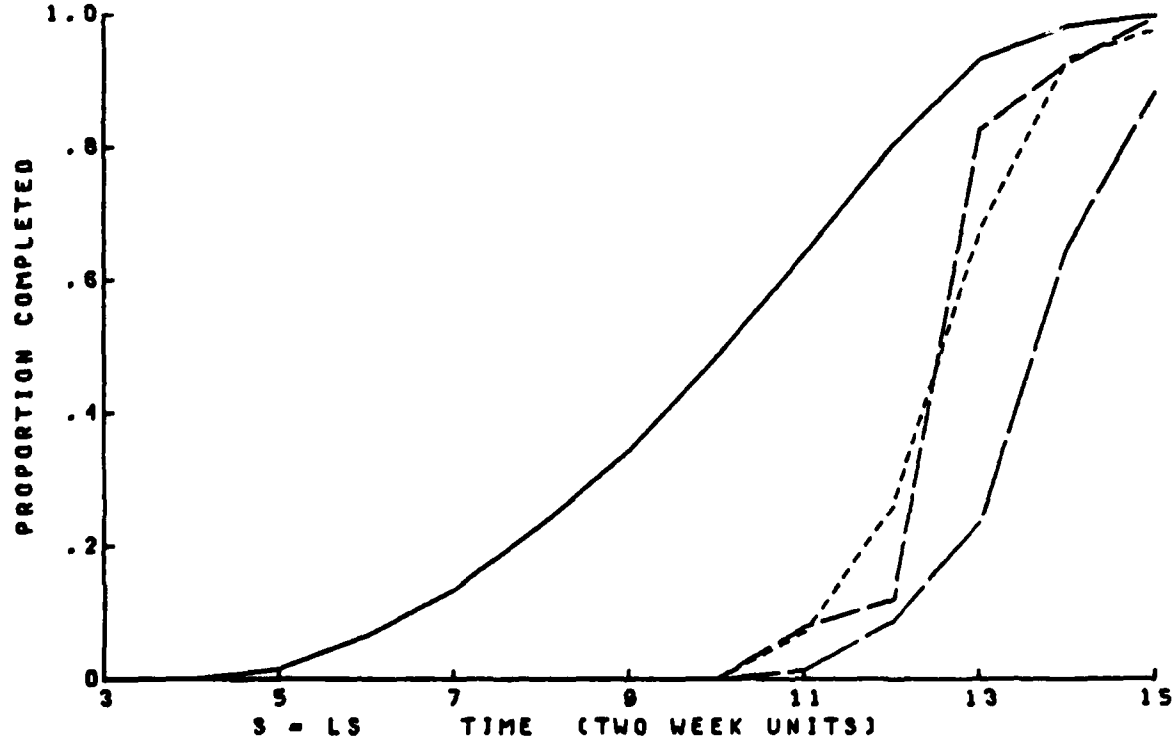


EXHIBIT 3.12

TIGHT OPERATION OF ACTIVITY WITH ITS PREDECESSOR

of A_j throughout its window, not only in the overlap.

In conclusion, the example from the overhaul data illustrates that intermediate product input and output is related to resource use in a complicated non-linear manner. The same conclusion holds for the relation between the operating modes of A_i and A_j when A_j operates tightly with A_i.

3.4.3. Relative Progress of Activity Operation

Let us suppose an activity A_i has a successor A_j, but that no other activity succeeds A_i or precedes A_j. Our objective is to describe the operating modes $OM_j=\{Z_j(t)\}_t$ that are consistent with a given operating mode $OM_i=\{Z_i(t)\}_t$ of A_i.

Suppose we could determine the operating mode of A_j when A_j operates tightly with the given mode OM_i of A_i; let us denote it by $OM_j^*=\{Z_j^*(t)\}_t$. The tight operating mode of A_j is the earliest feasible mode of A_j, given OM_i. An operating mode OM_j will be consistent with OM_i if, for all time periods, OM_j does not use more resources than OM_j^* does, i.e.,

$$Z_j(t) \leqslant Z_j^*(t) \qquad t=S_j, \ldots, F_j \ . \tag{3.9}$$

Thus, if we can determine OM_j^* in terms of OM_i, the operating modes OM_j consistent with OM_i can be described. Note that, at the key-op level, there may be many key-op schedules that induce an aggregate activity operating mode OM_i. This would imply that there is no unique OM_j^*, since each such schedule for A_i might induce a distinct tight operating mode for A_j. We will assume that the differences between such modes of A_j are insignificant.

In the last section we observed that OM_j^* is non-linearly related to OM_i. To obtain a workable approximation to the relationship between OM_i and OM_j^* we observe that, when A_i operates early ($OM_i=OM_i^E$) or late ($OM_i=OM_i^L$), OM_j^* is correctly presented, respectively, by OM_j^E or OM_j^L. In fact, we will require our approximation to satisfy these "boundary conditions." We relate OM_j^* to OM_i by comparing OM_i with its bounds OM_i^E and OM_i^L, and then requiring that OM_j^* must operate similarly with respect to its own bounds OM_j^E and OM_j^L.

Let us define $p_i(t)$, the *relative progress* of A_i at time t, as

$$p_i(t) \equiv \begin{cases} 0 & t=S_i \\ \dfrac{Z_i(t)-Z_i^L(t)}{Z_i^E(t)-Z_i^L(t)} & t=S_i+1,\ldots,F_i-1 \\ 1 & t=F_i \end{cases} . \tag{3.10}$$

We assume $Z_i^E(t)-Z_i^L(t)>0$ for $t=S_i+1,\ldots,F_i-1$. (Note that $Z_i^E(t)=Z_i^L(t)$ for $t=S_i$ and $t=F_i$.) Re-arranging (3.10), the operating mode OM_i can be expressed as

$$Z_i(t) = Z_i^L(t) + p_i(t)[Z_i^E(t)-Z_i^L(t)] \qquad t=S_i,\ldots,F_i \ . \tag{3.11}$$

We say that A_i operates early at t if $p_i(t)=1$, and operates late at t if $p_i(t)=0$. Note that the window curve constraints of Section 3.2.4 are equivalent to

$$0 \leqslant p_i(t) \leqslant 1 \qquad\qquad t=S_i, \ldots, F_i \ .$$

To compute $OM_j^*=\{Z_j^*(t)\}_t$ we equate its relative progress $\{p_j^*(t)\}_t$ to the relative progress of A_i, $\{p_i(t)\}_t$. It is not appropriate to do this with a simple time lag; rather, we will equate relative progress at corresponding points of the windows of A_i and A_j.

Let the window $(S_i,F_i]$ be partitioned into M subintervals as follows:

$$(S_iF_i] = (F_i^0,F_i^1] \cup \cdots \cup (F_i^{m-1},F_i^m] \cup \cdots \cup (F_i^{M-1},F_i^M] \ ,$$

where $F_i^0 \equiv S_i$ and $F_i^M \equiv F_i$. We denote the m^{th} subinterval $(F_i^{m-1},F_i^m]$ by $[m]_i$. The interior *partitioning points* $F_i^1, \ldots, F_i^{M-1}$ are determined by requiring that they divide the area between the window curves (i.e., the slack of A_i) into vertical slices of approximately equal size:

$$\sum_{t\in[m]_i} Z_i^E(t)-Z_i^L(t) \simeq \frac{1}{M} \sum_{t=S_i}^{F_i} Z_i^E(t)-Z_i^L(t) = \frac{\sigma_i}{M} \ .$$

We partition the window of activity A_j and the area between its window curves similarly, using the partitioning points F_j^m, $m=0,\ldots,M$. We relate $\{p_j^*(t)\}_t$ to $\{p_i(t)\}_t$ by equating the relative progress of each activity at the partitioning points:

$$p_j^*(F_j^m) = p_i(F_i^m) \qquad m=0,\ldots,M \ . \tag{3.12}$$

For interior subintervals $[m]_j$, $m=2,..,M-1$, the value $p_j^*(t)$ for each $t\in[m]_j$ is taken to be an interpolation of the values at the endpoints of $[m]_j$. Inside the exterior subintervals $[1]_j$ and $[M]_j$ we assume constant progress $p_j^*(F_j^1)$ and $p_j^*(F_j^{M-1})$. The transformation from $\{p_i(t)\}_t$ to $\{p_j^*(t)\}_t$ is thus completely determined by the following expression:

$$p_j^*(t) = \lambda_j^m(t)\,p_i(F_i^{m-1}) + (1-\lambda_j^m(t))\,p_i(F_i^m) \qquad t\in[m]_j; m=1,\ldots,M \ , \tag{3.13}$$

$$\textit{where} \qquad \lambda_j^m(t) = \begin{cases} 0 & m=1 \\ \dfrac{F_j^m - t}{F_j^m - F_j^{m-1}} & m=2,\ldots,M-1 \\ 1 & m=M \end{cases} \ .$$

The operating mode OM_j^* is then determined by

$$Z_j^*(t) = Z_j^L(t) + p_j^*(t)[Z_j^E(t) - Z_j^L(t)] \quad t = S_j, \ldots, F_j .$$

Note that the transformation preserves the desired correspondence between OM_i^E and OM_j^E, and between OM_i^L and OM_j^L.

Recall constraints (3.9) which state that OM_j is consistent with OM_i if

$$Z_j(t) \leqslant Z_j^*(t) \quad t = S_j, \ldots, F_j .$$

This is equivalent to

$$p_j(t) \leqslant p_j^*(t) \quad t = S_j, \ldots, F_j .$$

Using (3.12) we obtain the *relative progress transfer constraints* for the partitioning points:

$$p_j(F_j^m) \leqslant p_i(F_i^m) \quad m = 0, \ldots, M . \tag{3.14}$$

Including interior time points, and using (3.13), the constraints are given by

$$p_j(t) \leqslant \lambda_j^m(t)\, p_i(F_i^{m-1}) + (1 - \lambda_j^m(t))\, p_i(F_i^m) \quad t \in [m]_j ; m = 1, \ldots, M . \tag{3.15}$$

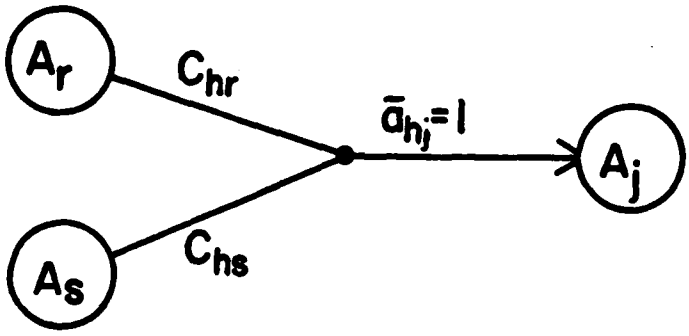
For this simple case of intermediate product transfer from A_i to A_j, OM_j is consistent with OM_i if its relative progress does not exceed the relative progress of A_i (with respect to the transformation (3.13) from $\{p_i(t)\}_t$ to $\{p_j^*(t)\}_t$). Similar statements can be made for all types of intermediate product transfers, as we demonstrate below.

3.4.4. Relative Progress Transfer Constraints

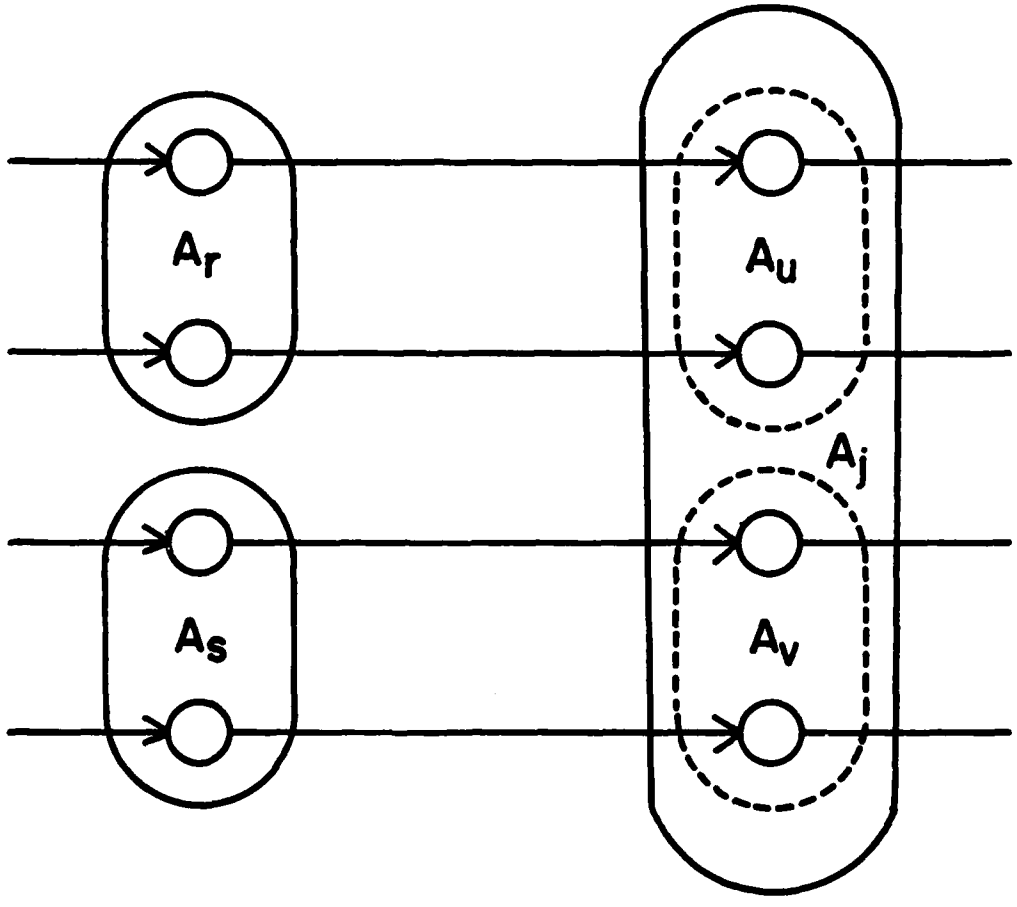
Recall from Section 3.4.1 that, in the general model, the intermediate product transfer constraints were given by

$$\sum_{j=1}^{N} \bar{a}_{hj} I_j(t) \leqslant \sum_{i=1}^{N} c_{hi} O_i(t) \quad h = 1, \ldots, H;\ t = 1, \ldots, T .$$

The constraints state that input of each intermediate product IP_h by time t may not exceed output by t. In this section we present analogous constraints stating that, at corresponding time points, the average relative progress of all users of IP_h may not exceed the average relative progress of the producers of IP_h:



(a) Flow of IP_h



(b) Underlying Key-op Subnetwork

EXHIBIT 3.13

MEASURING POOLED INTERMEDIATE PRODUCT OUTPUT

$$\sum_{j=1}^{N} \bar{a}_{hj} p_j(F_j^m) \leqslant \sum_{i=1}^{N} c_{hi} p_i(F_i^m) \quad h=1,...,H;\ m=0,...,M\ . \tag{3.16}$$

These constraints are discussed below for the three types of intermediate products introduced in Section 3.4.1, with appropriately chosen coefficients $\bar{a}_{hj}$ and c_{hi}.

(1) *Single Producer and User*

Suppose an activity A_i has more than one successor, and produces a distinct product for each of them. We assume that the production of each is identically distributed in relation to OM_i and therefore apply constraints (3.14) to each intermediate product output IP_h of A_i. Letting $c_{hi}=1$ and $\bar{a}_{hj}=1$, the following holds for IP_h:

$$\bar{a}_{hj} p_j(F_j^m) \leqslant c_{hi} p_i(F_i^m) \Leftrightarrow p_j(F_j^m) \leqslant p_i(F_i^m) \quad m=0,...,M\ . \tag{3.17}$$

For time periods within the partitiioning intervals we interpolate relative progress as in (3.15).

Constraints (3.17) imply that, if an activity has multiple predecessors and receives distinct intermediate products from them, its operation is constrained by the relative progress of each of them.

(2) *Single User, Pooled Production*

Suppose an activity A_j uses an intermediate product IP_h that is produced by several activities. In other words, predecessors of A_j pool their output for use by A_j. We state that use of IP_h is consistent with its production if the relative progress of A_j does not exceed the "average" relative progress of the producers of IP_h. At the partitioning points these constraints are stated as follows:

$$p_j(F_j^m) \leqslant \sum_{i=1}^{N} c_{hi} p_i(F_i^m) \quad m=0,...,M\ , \tag{3.18}$$

where the output coefficients c_{hi} must still be specified. For time periods within the intervals $[m]_j$, we interpolate relative progress as before. The details are omitted here.

We motivate the use of constraints (3.18) by giving a simple example shown in Exhibit 3.13(a). The activities A_r and A_s pool their output for use by A_j. Recall that a key-op subnetwork similar to that in Exhibit 3.13(b) underlies Exhibit 3.13(a). All successors of key-ops within A_r and A_s were grouped into A_j.

Suppose two aggregate activities A_u and A_v had been formed instead, as indicated by the dashed lines in Exhibit 3.13(b). Then A_u succeeds A_r and A_v succeeds A_s. Note that A_u and

A_v are assumed to have similar work content, and that the following holds:

$$a_j Z_j(t) = a_u Z_u(t) + a_v Z_v(t) \quad t=S_j, \ldots, F_j \ . \tag{3.19}$$

The coefficients a_j, a_u, and a_v are the total resource requirements of each activity, and $a_j=a_u+a_v$.

Let us consider the transfers of IP_h in terms of the relative progress transfers from A_r to A_u, and from A_s to A_v. Given OM_r and OM_s, the operating modes OM_u^* and OM_v^* for which A_u operates tightly with A_r, and A_v with A_s, are determined as in (3.12):

$$\begin{aligned} p_u^*(F_u^*) &= p_r(F_r^m) \\ p_v^*(F_v^*) &= p_s(F_s^m) \end{aligned} \qquad m=0,\ldots,M \ . \tag{3.20}$$

Our goal is to obtain an expression for OM_j^*, the operating mode of A_j when it operates tightly with its predecessors A_r and A_s. From (3.19) we know that

$$Z_j^*(t) = \frac{a_u}{a_j} Z_u^*(t) + \frac{a_v}{a_j} Z_v^*(t) \quad t=S_j, \ldots, F_j \ .$$

It can be shown that, if the window curves of A_u and A_v have the same shape (which implies that the window curves of A_j also have this shape), then the following also holds:

$$p_j^*(t) = \frac{a_u}{a_j} p_u^*(t) + \frac{a_v}{a_j} p_v^*(t) \quad t=S_j,\ldots,F_j \ . \tag{3.21}$$

In practice this condition will usually only be approximately satisfied.

By combining (3.20) and (3.21) we obtain relative progress transfer constraints on the operation of A_j:

$$p_j(F_j^m) \leqslant \frac{a_u}{a_j} p_r(F_r^m) + \frac{a_v}{a_j} p_s(F_s^m) \quad m=0,\ldots,M \ .$$

Comparing these constraints with (3.18) we see that, in general, the output coefficient c_{hl} should be the proportion of resources used by A_j to perform those key-ops which succeed key-ops within its predecessor A_l.

(3) *Single Producer, Shared Use*

If intermediate product output IP_h of activity A_i is shared by several successors of A_i, the constraints are as follows:

$$\sum_{j=1}^{N} \bar{a}_{hj} p_j(F_j^m) \leqslant p_i(F_i^m) \qquad m=0,...,M \ .$$

The coefficient $\bar{a}_{hj}$ is taken to be the proportion of resources used by A_i to perform those key-ops which precede key-ops within successor A_j. The motivation for the above constraints is similar to that of case (2). Note, however, that tight operation of one successor A_j with A_i implies assumptions on the operation of all successors of A_i that share IP_h. It is therefore easier in this case to interpret tight operation of the successors with A_i in the opposite direction. For given operating modes of the successors we would seek the "latest" possible operating mode of A_i. The interpolation of relative progress transfer within the partitioning intervals is best represented backwards, also.

3.5. Attainable Window Curves: Workspace Limitations

3.5.1. Workspace Limitations

In Sections 3.2.3 and 3.2.4 we developed the following bounds on the operating mode of each activity A_i:

$$\begin{array}{l} 0 \leqslant z_i(t) \leqslant \bar{z}_i(t) \\ Z_i^L \leqslant Z_i(t) \leqslant Z_i^E(t) \end{array} \qquad t = S_i,\ldots,F_i \ . \tag{3.22}$$

The bounds $\bar{z}_i(t)$, $Z_i^L(t)$, and $Z_i^E(t)$ were computed by summing the operating intensities of all key-ops within A_i that could operate at time t. Although each of these key-ops *can* operate at t, they cannot necessarily operate *together* at t. Workspace limitations (interferences) may exist that restrict the operation of the activity. As an example, there may be insufficient space for the concurrent removal of all engine room components.

The workspace that an activity A_i needs can be considered a resource that is unique to the activity. We express the limitation on the availability of this resource as an upper bound, $\hat{z}_i$, on the operating intensity of A_i:

$$z_i(t) \leqslant \hat{z}_i \qquad t=S_i,\ldots,F_i \ . \tag{3.23}$$

Other types of resources that are unique to an activity can be treated similarly.

The early and late operating modes of activity A_i do not necessarily satisfy (3.23). For example, suppose A_i is preceded by a restraining key event. When A_i operates early, all key-ops within A_i start operating immediately after the key event and their combined intensities could exceed $\hat{z}_i$.

Let A_j be a successor of A_i. If the early operating mode of A_i does not satisfy (3.23) then A_i cannot transfer intermediate product output to A_j as fast as was originally assumed. Therefore the early operating mode of A_j is infeasible, and the workspace limitations on A_i should be *propagated* onto the early operating mode of A_j in some way. The limitations on A_i must also be propagated (indirectly) to successors of A_j, and to their successors, etc.

In general, workspace limitations must be propagated throughout the activity network. If the early operating mode of an activity violates (3.23) then the limitation is propagated forward through the network; if the late mode violates (3.23) then it is propagated backward. Note that the forward and backward propagations are independent of each other. The early and late

operating modes that result from this propagation of workspace limitations throughout the network are referred to as being *attainable*.

In Section 3.7.2 we will see that it is important to replace the original early and late modes with the attainable ones. These are computed in the next section. The constraints (3.22) and (3.23) will then always be satisfied for the early and late modes, but we must still explicitly restrict all other operating modes to satisfy them, and also the relative progress transfer constraints of Section 3.4.3.

3.5.2. Propagation of Workspace Limitations

For each activity A_i, let $\hat{OM}_i^E = \{\hat{Z}_i^E(t)\}_t$ and $\hat{OM}_i^L = \{\hat{Z}_i^L(t)\}_t$ be the attainable early and late operating modes of A_i. We could compute them by performing simulations on the key-op network subject to the following constraints on key-op start time schedules S:

$$\sum_{l \in \Lambda(S,t)} \frac{b_l}{a_i} \bar{z}_l \leqslant \hat{z}_i \qquad i=1,\ldots,N;\ t=S_i, \ldots, F_i ,$$

where $\Lambda(S,t)$ is the index set of key-ops within A_i that can operate at t, given S. We would need to develop considerable programming logic (resource allocation rules, activity priority rules) to perform these simulations. As an alternative approach, we will approximate the attainable modes by working within the aggregate model.

Let A_j be a successor of activity A_i and assume, for simplicity of exposition, that A_j has no predecessors other than A_i. Given $\hat{OM}_i^E$, we compute $\hat{OM}_j^E$ as follows:

(1) Determine $OM_j^* = \{Z_j^*(t)\}_t$, the operating mode of A_j when it operates tightly with $\hat{OM}_i^E$ (as in Section 3.4.3).

(2) Modify OM_j^* to reflect the workspace limitations on A_j. This is done by computing the attainable cumulative intensity $\hat{Z}_j^E(t)$ recursively for $t=S_j+1,\ldots,F_j$:

$$\hat{Z}_j^E(t) = \min \{\hat{Z}_j^E(t-1) + \hat{z}_j ,\ Z_j^*(t)\} , \tag{3.24}$$

where $\hat{Z}_j^E(S_j) \equiv 0$, and $Z_j^*(t)$ is computed as in Section 3.4.3.

The computation is illustrated in Exhibit 3.14. The curve $Z_j^*(t)$ is followed until, for some t, its slope exceeds $\hat{z}_j$. We then extend the new curve with slope $\hat{z}_j$ until the curve $Z_j^*(t)$ is reached again. Note that, if OM_j^* satisfies the workspace limitations on A_j, then $\hat{OM}_j^E = OM_j^*$ and hence $\hat{OM}_j^E$ tightly follows $\hat{OM}_i^E$. Also, if A_j has no predecessors, $\hat{OM}_j^E$ can be computed

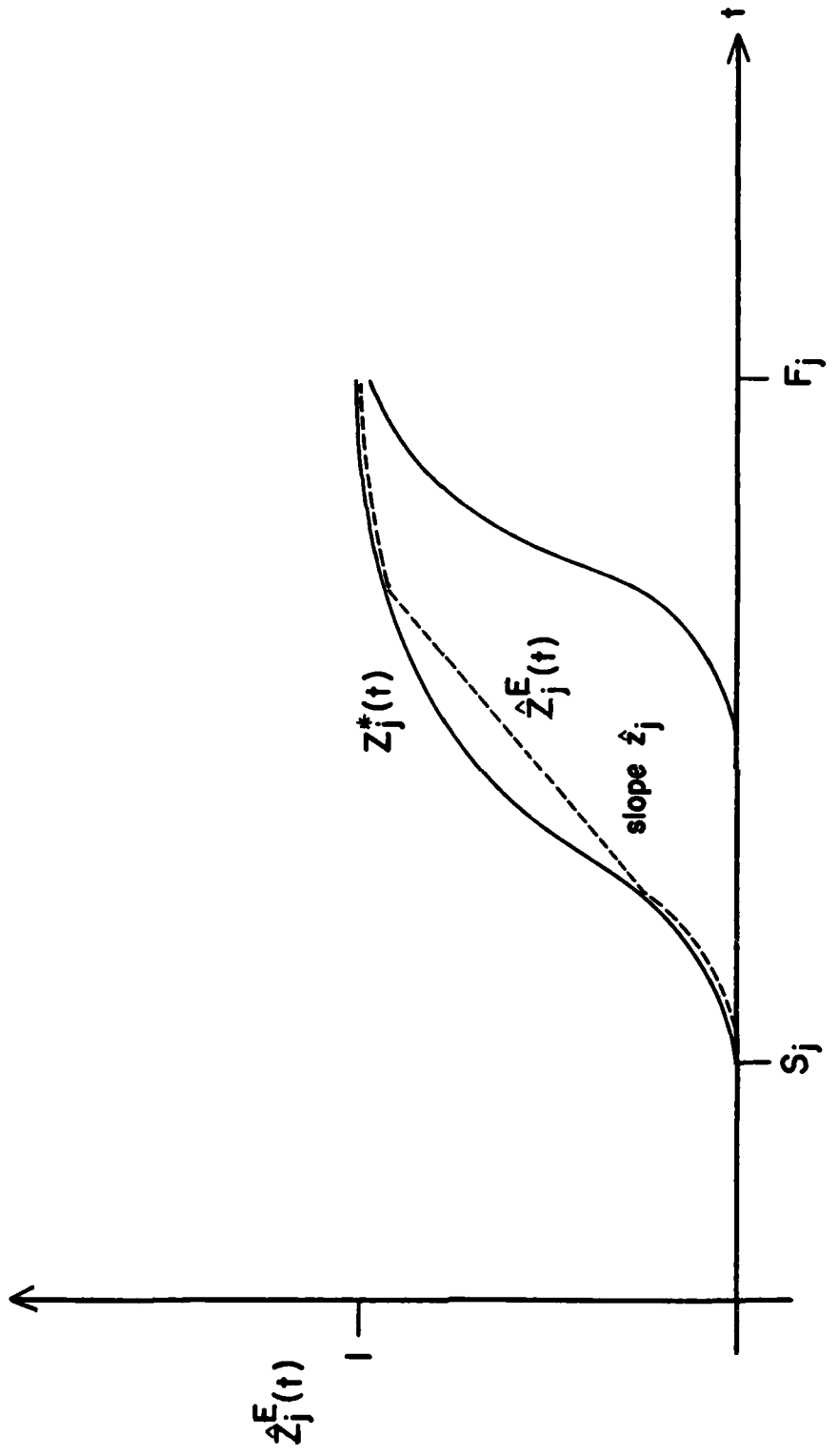


EXHIBIT 3.14

PROPAGATION OF WORKSPACE LIMITATIONS

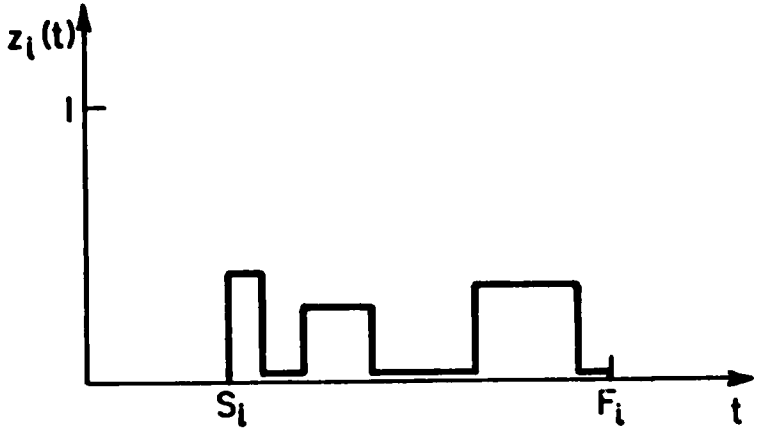
by using (3.24) with OM_j^* replaced by OM_j^E, the original early operating mode of A_j.

The attainable early operating modes are computed by starting with those activities with no predecessors. Then, proceeding forward through the network, we apply (3.24) successively. If an activity has multiple predecessors then (3.24) must be modified appropriately. If an activity shares its intermediate product output with multiple successors we preallocate the output (as represented by the activity's relative progress) equally to the successors. These modifications are easily made by considering the different types of relative progress transfer constraints discussed in Section 3.4.4.

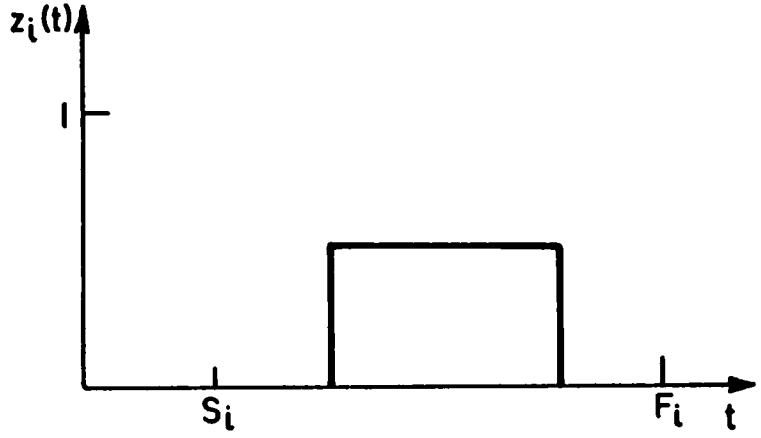
In a completely analogous manner we can propagate workspace limitations backward through the network, the details of which are omitted. We will assume that the computations of this section have been applied to the original window curves of all activities, so that they are attainable. Relative progress is measured from now on with respect to these attainable curves. In general, the relative progress transfers that result when using the original versus the attainable curves will be different. There are conditions, however, under which the transfers are the same.

These conditions frequently arise is practice. Consider a series of three activities representing removal (RMV), repair (RPR), and reinstallation (RI) of a technical system. Suppose that the removal activity RMV is preceded by a restraining key event, and RI is succeeded by a restraining key event. Since RMV succeeds a key event, its early mode is quite peaked (as mentioned before in Section 3.5.1) and may violate its workspace limitation. The early modes of RPR and RI are much less peaked since the key-ops within them start in a staggered manner. Hence the early modes of RPR and RI may not violate their limitations. Analogously, RI is the only activity whose late mode is likely to violate its workspace limitation.

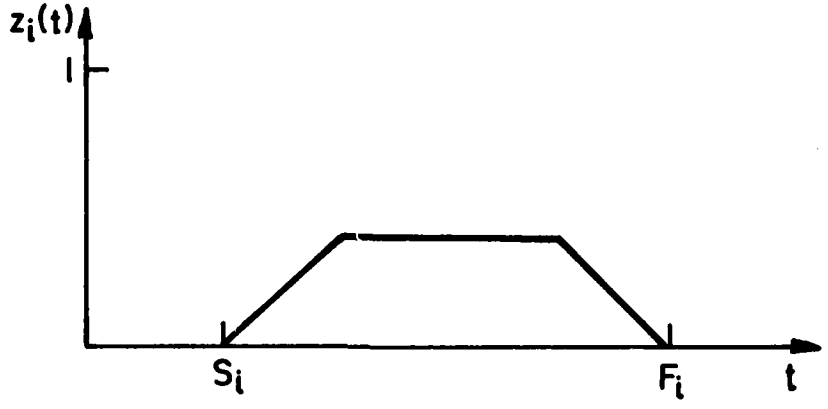
Under these conditions the set of consistent operating modes will be approximately the same whether relative progress is measured with respect to the original or the attainable window curves. This is true as a direct consequence of the way in which workspace limitations are propagated. The qualifier "approximately equal" applies since we interpolate relative progress for the time periods between window partitioning points (see Section 3.4.3).



(a) Unrealistic



(b) Constant Intensity



(c) Trapezoidal

EXHIBIT 3.15

EXAMPLES OF ACTIVITY OPERATING MODES

3.6. Modeling Activity Operation - Computational Issues

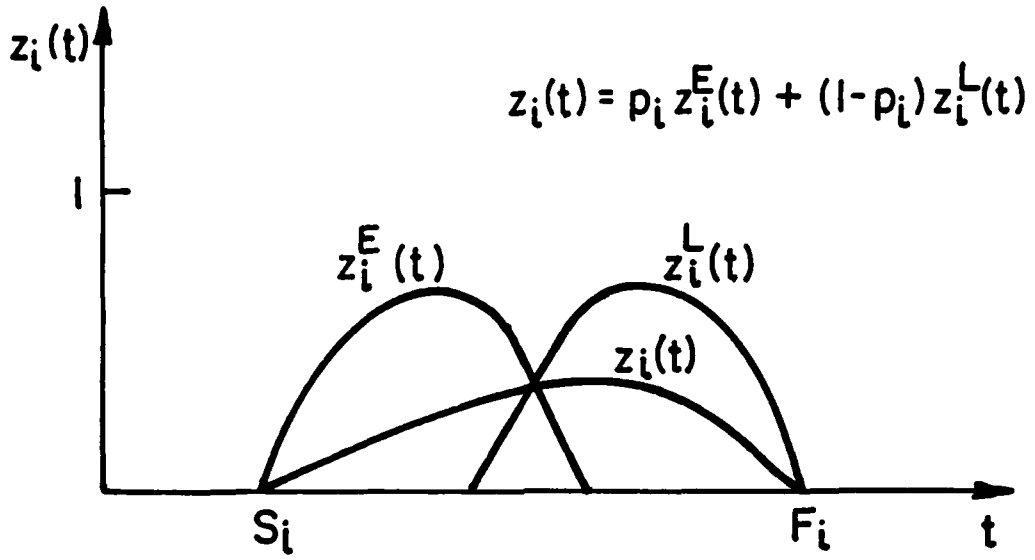
The aggregate overhaul model is part of an *LP* formulation of shipyard resource allocation described in Chapter 4. Before presenting the final version of the model in Section 3.7, we first discuss some computational issues relating to the realism of activity operating modes and the number of decision variables involved.

Suppose the operating intensities $z_i(t)$, $t=S_i+1,...,F_i$, are taken as the decision variables for operation of each activity A_i. The *activity operation constraints* developed thus far (intensity bounds, window curve constraints, relative progress transfer constraints) do not prevent an operating mode from being obtained that is obviously unrealistic. As an example, consider Exhibit 3.15(a). The properties of basic solutions to the *LP* model may even induce such unrealistic operating modes. Furthermore, the resulting number of variables and constraints in the *LP*, while not impossible to accommodate, is nevertheless quite large.

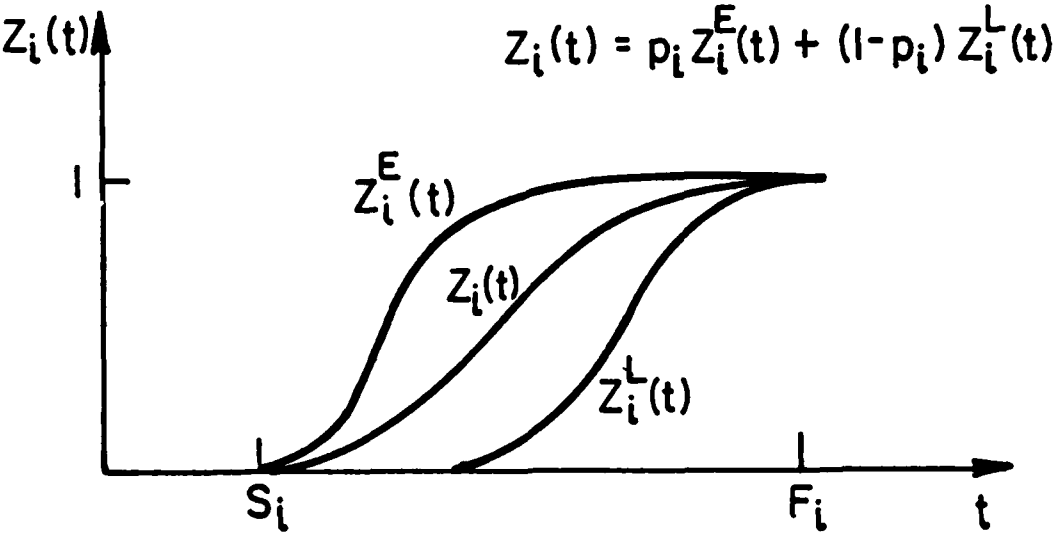
For these reasons, we will characterize the operating modes in the next section by fewer decision variables in a way that more realistic solutions are induced. To motivate our choice of decision variables let us discuss a few simple types of activity operating modes:

(1) *Constant Intensity:* The activity is assumed to operate at a constant, but variable, rate over a subinterval of $(S_i, F_i]$, as illustrated in Exhibit 3.15(b). The decision variables would be the activity start time and the intensity level. This type of operating mode is not easily reconciled with the activity window curves, and does not reflect aggregate product transfer characteristics. Furthermore, it cannot be represented by linear constraints and continuous variables.

(2) *Constant Relative Progress:* In this case a relative progress variable p_i applies over the whole window of A_i. See Exhibit 3.16 for an example of the resulting intensity curve $z_i(t)$ and cumulative curve $Z_i(t)$. Note that this type of operating mode is a convex combination of the window curves. From the development of Section 3.4.3, we know that it will allow intermediate product transfers to be modeled effectively, but the results obtained may be undesirable. For example, if $z_i^E(t)=z_i^L(t)=c$ for some t then we are forced to have $z_i(t)=c$ for all operating modes. This is obviously too restrictive. If $c=0$, then all operating modes will have an interval with zero intensity in the middle of the window, which is highly unrealistic.



(a) Intensity Curves



(b) Cumulative Intensity Curves

EXHIBIT 3.16

CONSTANT RELATIVE PROGRESS OPERATING MODE

(3) *Trapezoidal Shape:* In practice, an aggregate activity does not use resources at a constant rate. Rather, there is a start-up phase during which resource use slowly increases, then a fairly steady phase of operation, and finally a phase in which resource use decreases. Exhibit 3.15(c) illustrates this trapezoidal type of operating mode. In the next section, we show that such operating modes can be approximated in terms of linear constraints and real variables, while still retaining the characteristics of relative progress transfer.

3.7. Three Phase Model of Activity Operation

In the last section we outlined the following desirable properties of a computational model of activity operation:

- satisfaction of the activity operation constraints
- few variables and constraints
- "reasonable" operating modes, e.g., trapezoidal shape.

The *three phase model* presented below is an attempt to satisfy these properties without making too many compromises.

3.7.1. Phase Variables

Let us partition the window $(S_i, F_i]$ of each activity A_i into three phases, which we call the first, middle, and last phases. The partition is defined by

$$(S_i, F_i] = (S_i, F_i^1] \cup (F_i^1, F_i^2] \cup (F_i^2, F_i] .$$

We define one decision variable for each phase of activity A_i. The operating mode is then represented in terms of these *phase variables.* In the middle phase, we choose to hold operating intensity constant, and denote the variable for intensity during the phase by $z_i[2]$. Instead of intensity, it is the *relative progress* which we hold constant over each of the other phases. Let $p_i[1]$ and $p_i[3]$ be the variables of the first and last phases.

The operating mode $\{Z_i(t)\}_t$ can thus be represented by:

$$Z_i(t) = \begin{cases} Z_i^L(t) + p_i[1](Z_i^E(t) - Z_i^L(t)) & S_i < t \leqslant F_i^1 \\ Z_i(F_i^1) + (t - F_i^1) z_i[2] & F_i^1 < t \leqslant F_i^2 \\ Z_i^L(t) + p_i[3](Z_i^E(t) - Z_i^L(t)) & F_i^2 < t \leqslant F_i \end{cases} \tag{3.25}$$

Exhibit 3.17 shows the resulting type of cumulative intensity curve. This construction induces

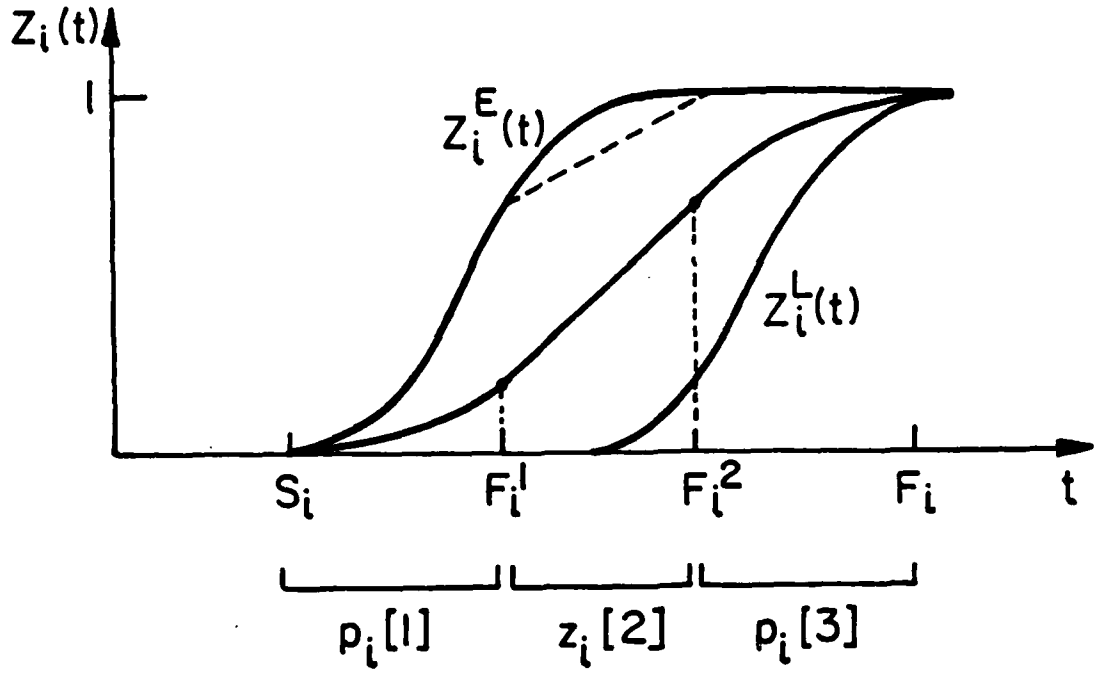


EXHIBIT 3.17

THREE PHASE OPERATING MODE

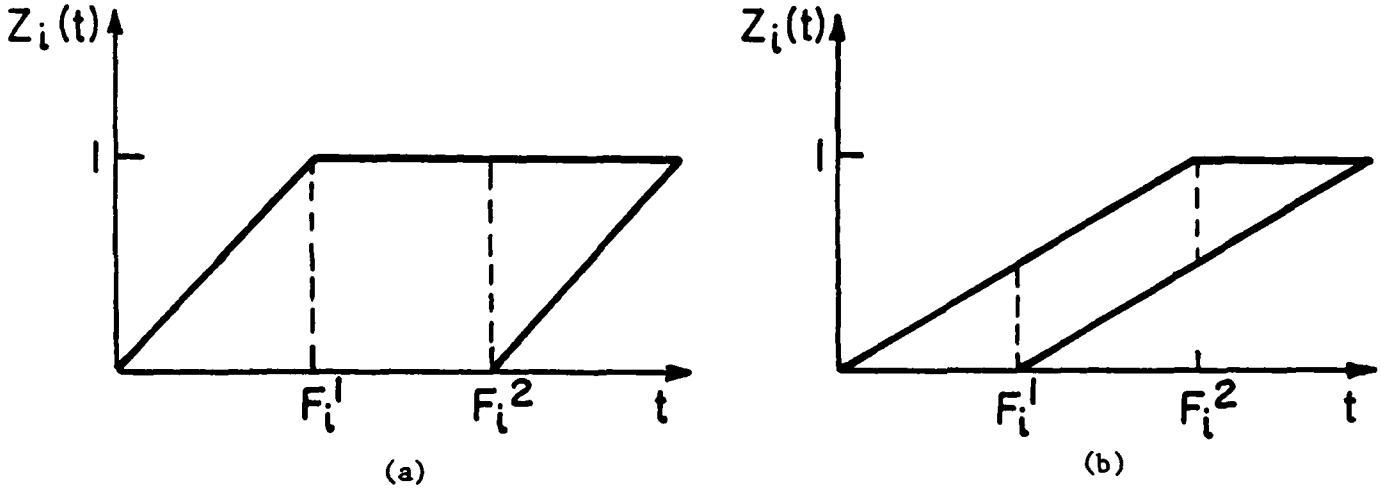


EXHIBIT 3.18

EXAMPLES OF WINDOW CURVES PARTITIONING

an intensity curve which increases at first, is then constant, and finally decreases again. (While the resulting operating mode is not trapezoidal, it is a fairly good approximation.)

To ensure that the curve is continuous, we must place a boundary condition on the operation of A_i at time $t=F_i^2$, namely

$$p_i(F_i^2) = p_i[3] \ , \tag{3.26}$$

which is equivalent to

$$Z_i(F_i^2) = Z_i^L(F_i^2) + p_i[3](Z_i^E(F_i^2) - Z_i^L(F_i^2)) \ . \tag{3.27}$$

Noting that $z_i(t)=Z_i(t)-Z_i(t-1)$, we can use (3.25) and (3.27) to obtain the following expression for $z_i(t)$:

$$z_i(t) = \begin{cases} z_i^L(t) + p_i[1](z_i^E(t) - z_i^L(t)) & S_i < t \leqslant F_i^1 \\ z_i[2] & F_i^1 < t \leqslant F_i^2 \\ z_i^L(t) + p_i[3](z_i^E(t) - z_i^L(t)) & F_i^2 < t \leqslant F_i \end{cases} . \tag{3.28}$$

Recall from Section 3.6 that if relative progress is constant throughout the window and, for some t, $z_i^E(t)=z_i^L(t)=c$, then $z_i(t)=c$ for all operating modes. Equivalently, if the slopes of $Z_i^E(t)$ and $Z_i^L(t)$ are equal then $Z_i(t)$ must have that slope. To avoid prespecifying the slope it may be necessary to modify the criterion for partitioning the window into phases (which was presented in Section 3.4.3). Instead of partitioning the area between the window curves into equal thirds, the middle phase should be widened to include all time periods in which the slope of the window curves is equal. Exhibit 3.18 shows two stylized examples of this.

Up to now we have implicitly assumed that resource use by the aggregate activities is measured in terms of the time units of key-op resource use, namely days. But shipyard resource allocations must be made for larger time intervals. Hereafter we assume that time intervals are periods of two work weeks (ten working days). The computation of the activity operating modes is performed on this wider grid. Clearly, some information is thereby lost, but daily resource use cannot, and should not, be represented in an aggregate model. Resource use by activity A_i during each two week period $(t-1, t]$ is given by

$$x_{ki}(t) = a_{ki} z_i(t) \quad k=1,...,K \ ,$$

where $z_i(t)$ is computed according to (3.28).

3.7.2. Activity Operation Constraints

In this section we discuss the precise form of the constraints on the phase variables of each activity.

Intensity Bounds

For all $t=S_i, \ldots, F_i$, the operating intensities of activity A_i, as given by (3.28), must satisfy

$$0 \leqslant z_i(t) \leqslant \bar{z}_i(t) \tag{3.29}$$

$$z_i(t) \leqslant \hat{z}_i \ . \tag{3.30}$$

First, let us consider the middle phase. The upper bound $\bar{z}_i(t)$ is usually so ineffective that we can ignore it. What we do require is that

$$0 \leqslant z_i[2] \leqslant \hat{z}_i \ . \tag{3.31}$$

It may also be necessary to place a non-zero lower bound on $z_i[2]$ in order to reflect labor crew indivisibilities. Note that the maximum amount that could be performed during the middle phase is given by $Z_i^E(F_i^2) - Z_i^L(F_i^1)$. The upper bound in (3.31) is therefore redundant if the following condition holds:

$$Z_i^E(F_i^2) - Z_i^L(F_i^1) \leqslant \hat{z}_i \ .$$

In the first and last phases we observe that $z_i(t)$, as given in (3.28), satisfies (3.29) by definition (see Section 3.2.3), and satisfies (3.30) by construction (the window curves are assumed to be attainable).

Window Curve Constraints

The cumulative intensity $Z_i(t)$ of activity A_i must satisfy

$$Z_i^L(t) \leqslant Z_i(t) \leqslant Z_i^E(t) \qquad t = S_i, \ldots, F_i \ . \tag{3.32}$$

In the first and last phases these constraints become simple upper and lower bounds on the phase variables:

$$0 \leqslant p_i[m] \leqslant 1 \qquad m=1,3 \ .$$

Relating the constant intensity in the middle phase to the window curves would be more difficult. Note that the cumulative intensity curve for the early operating mode is usually concave in the middle phase, and for the late mode it is usually convex. Assuming that this is the case, an operating mode will satisfy (3.32) in the middle phase whenever

- the operating intensity is constant throughout the middle phase and
- the operating mode is within the window curves in the first and last phase.

(The dashed line in Exhibit 3.17 illustrate this argument.) Since these conditions are satisfied by construction, we need not explicitly require (3.32) to hold in the middle phase. As a consequence, however, the possible operating modes of an activity are restricted in the three phase model so that the early and late modes can only be approximately achieved.

Relative Progress Transfer Constraints

The constraints (3.16), when applied to the first and last phases, become

$$\sum_{j=1}^{N} \bar{a}_{hj} p_j[m] \leqslant \sum_{i=1}^{N} c_{hi} p_i[m] \qquad h=1,\ldots,H\ ;\ m=1,3\ .$$

We ignore the progress transfer constraints in the middle phase. We believe that the small loss in accuracy is not worth the difficulty of enforcing such constraints in this phase.

Summary

The operation of each activity A_i is represented by three phase variables $p_i[1]$, $z_i[2]$, $p_i[3]$. The operating mode is a function of these variables, as shown in (3.28). The constraints on activity operation are summarized below:

$$\begin{aligned} 0 \leqslant z_i[2] &\leqslant \hat{z}_i && i=1,\ldots,N \\ 0 \leqslant p_i[m] &\leqslant 1 && m=1,3;\ i=1,\ldots,N \\ \sum_{j=1}^{N} \bar{a}_{hj} p_i[m] &\leqslant \sum_{i=1}^{N} c_{hi} p_i[m] && m=1,3;\ h=1,\ldots,H\ . \end{aligned}$$

Note that the boundary condition (3.26) in the middle phase becomes

$$\begin{aligned} [Z_i^E(F_i^1)-Z_i^L(F_i^1)]\,p_i[1] + (F_i^2-F_i^1)\,z_i[2] - [Z_i^E(F_i^2)-Z_i^L(F_i^2)]\,p_i[3] \\ = Z_i^L(F_i^2)-Z_i^L(F_i^1)\ . \end{aligned} \tag{3.33}$$

By using this relationship to express one of the phase variables in terms of the others, we can

reduce the number of variables in the problem. However, each such substitution would increase the number of inequalities that cannot be represented as simple upper bounds on variables.

3.7.3. Adaptation of Three Phase Model to Overhaul Data

The three phase model was adapted to the aggregate network constructed from overhaul data (see Section 3.3.4). Two modifications were made, however. Referring to Exhibit 3.9, they are described below:

(1) Activities A_1, A_{24}, and A_{39} operate with so little slack and within such narrow windows that only one variable per activity was defined. (Relative progress was held constant throughout each activity's window.) Furthermore, A_{32} has so little slack that its operating mode was held fixed.

(2) The overlap between engine room repair and reinstallation activities ($A_{25}, A_{28}, A_{30}, A_{31}$) and the succeeding test activity (A_{29}) was so short that relative progress was transferred from the last phase of the predecessors to the first phase of A_{29}. The same modification applies to the boiler room activities.

Using (3.33), the middle phase intensity variables were substituted for relative progress phase variables. As a result, 110 activity operation constraints in 73 variables, plus upper bounds, were obtained. Workspace limitations were imposed in some cases, and propagated through the network (as described in Section 3.5).

4. SHIPYARD RESOURCE ALLOCATION MODEL

In Chapter 1 we identified the need to improve aggregate planning in the shipyard. Labor productivity can be increased by making the allocation of labor resources to overhauls more explicit and consistent. We present a linear programming (LP) model that helps achieve this goal.

In Section 4.1 the aggregate planning process is discussed from two perspectives: overhaul planning and labor resource planning. The LP model for resource allocation is described in Section 4.2. We suggest ways of updating overhaul data in the aggregate model in Section 4.3. Finally, computational examples are presented in Section 4.4.

4.1. Aggregate Planning Process

Exhibit 4.1 shows a simple view of aggregate planning in the shipyard. Overhaul and labor shop information are combined in order to allocate shipyard labor resources. Labor use plans for each overhaul and utilization plans for each shop are thereby derived. We discuss the planning process first from the viewpoint of overhaul planning.

An overhaul execution plan is not made once and for all, but evolves dynamically through time. It is made more specific as information on the overhaul's work content becomes available. At a detailed level, the execution of the overhaul can be described as a network of key-ops and a schedule for their performance. However, the planning process must be able to deal with changes in labor allocations due to the competition for labor by ongoing and future overhauls.

We therefore place detailed overhaul planning in the framework shown in Exhibit 4.2. Detailed information is aggregated and combined with that of all other overhauls. An aggregate execution plan is derived and then disaggregated. The aggregate plan consists of a labor use plan for each shop and a timetable of key events. This framework is not intended to show the progression of planning steps through time. Rather, it shows the continuous coordination of planning at different levels.

To describe the framework it is useful to follow the development of the execution plan from an overhaul's conception through its completion. When it is first conceived, rough estimates of labor use are made. A preliminary aggregate overhaul model is developed from historical information. Tentative key event timetables are then considered; in particular, trial dates are established for important restraining events such as docking and undocking.

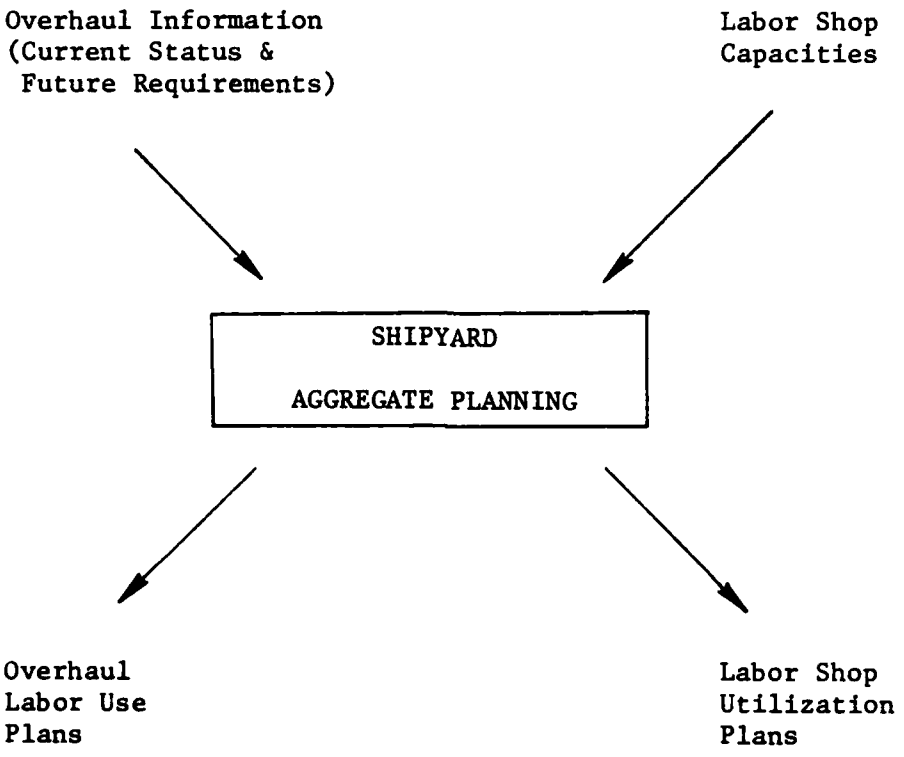


EXHIBIT 4.1

SHIPYARD AGGREGATE PLANNING

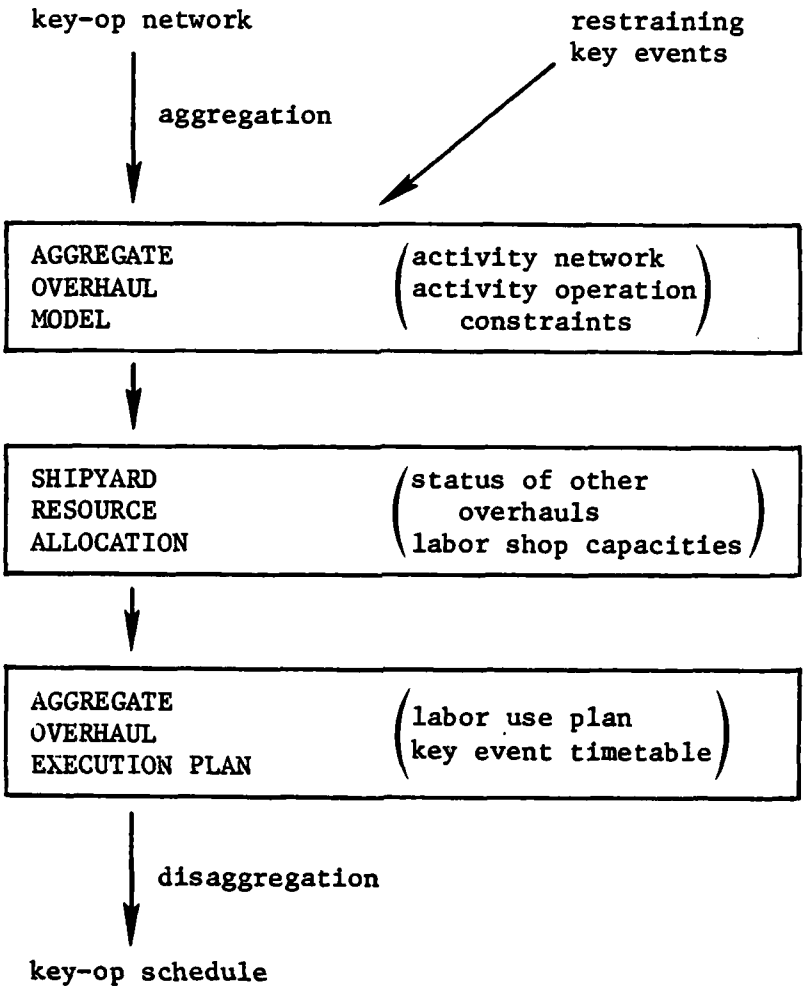


EXHIBIT 4.2

OVERHAUL PLANNING FRAMEWORK

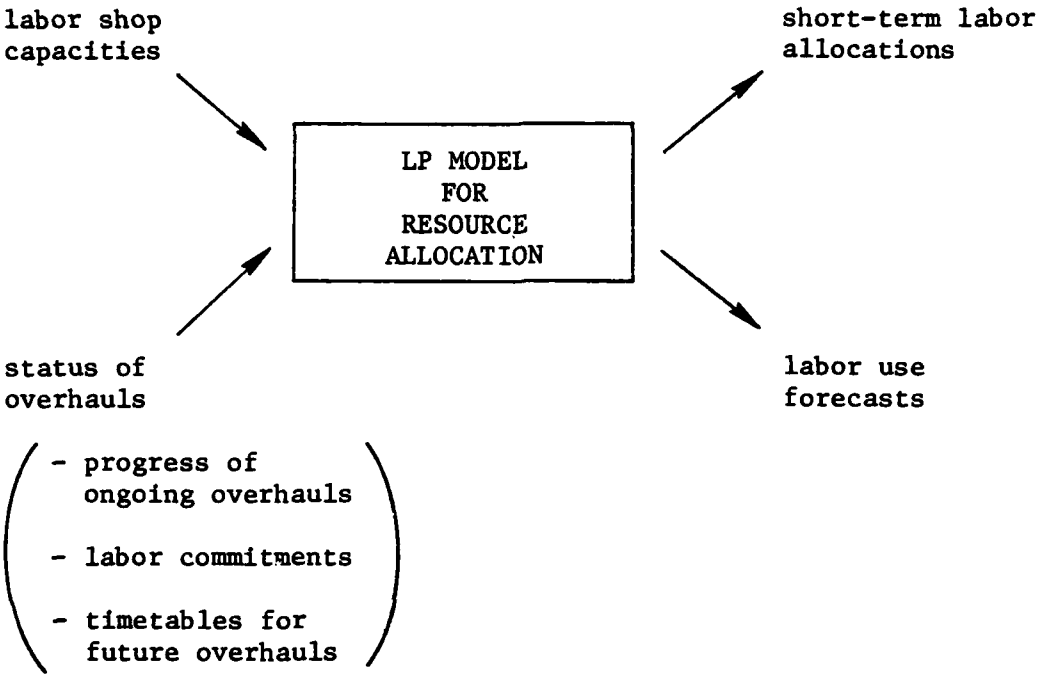


EXHIBIT 4.3

SHIPYARD RESOURCE ALLOCATION MODEL

Alternative labor use plans can be studied by experimenting with different timetables.

At the same time, detailed overhaul planning (as described in Section 1.2) takes place. By using this information, the aggregate overhaul model can be updated and refined. Once labor allocations to the overhaul are established in aggregate planning, scheduling can begin. When the overhaul is started, a specific schedule is required for the short-term only. Clearly, labor allocations and schedules for more distant time periods are tentative. The scheduling network for later stages of the overhaul need not be fully developed.

During the execution of the overhaul, labor use is monitored. The progress of the aggregate activities can be measured in terms of labor application versus estimated total requirements. Labor use for the near future is likely to be partially preallocated due to previous work crew assignments. We refer to these preallocations as labor *commitments.* As the overhaul proceeds, the flexibility of allocating labor for future time periods will decline until all future labor use has been committed.

Let us now turn to the other aspect of aggregate planning, namely resource planning. At regular intervals the utilization of labor in the shipyard is re-evaluated. The status of all overhauls is updated, there is an extension of the horizon up to which labor use is studied, and current and future labor shop capacities are revised.

The LP model described in the next section finds the most efficient utilization of shop capacities while taking into account the timely completion of all overhauls. Simply put, we try to "pack" the shipyard with work. Exhibit 4.3 summarizes the inputs and outputs of the model. (In Section 4.3 we discuss the updating of overhaul status in more detail.) The model provides plans for short-term allocations of labor among overhauls and forecasts of future labor requirements. These short-term allocations become labor commitments when the analysis is repeated in subsequent time periods.

4.2. LP Model of Resource Allocation

The problem of determining the most efficient allocation of resources (labor shop capacities) to overhauls is formulated as a linear program. We characterize the model in terms of its variables and constraints, and discuss its objective.

Variables

Let us denote the ongoing and future overhauls in the shipyard by $OH_1, \ldots, OH_J$. The phase variables of each activity A_i within an overhaul OH_j determine the operating mode $\{z_i(t)\}_t$ of A_i, and hence the load histories $\{x_{ki}(t)\}_t$ of A_i for each labor resource R_k. The amount of R_k applied to OH_j during $(t-1, t]$ is denoted by $x_k^j(t)$, so that

$$x_k^j(t) = \sum_{A_i \in OH_j} x_{ki}(t) = \sum_{A_i \in OH_j} a_{ki} z_i(t) \ .$$

Although the activity phase variables are the actual LP model variables, it is convenient to express resource use in terms of the linear functions $x_k^j(t)$ of these variables. We restrict our attention to resource use within time periods $(t-1, t]$, $t=1, \ldots, T$, and denote the allocation of R_k to OH_j during $(O, T]$ by $\{x_k^j(t)\}_t$. The overhauls do not necessarily operate only within $(O, T]$, however. If an overhaul finishes beyond T then not all of its activities will be included in the model. If an overhaul is in progress at time zero, some phase variables will be predetermined.

Constraints

Activity Operation Constraints: These constraints (summarized in Section 3.7.2) describe the set of consistent activity operating modes in terms of the phase variables. Overhaul resource allocations $\{x_k^j(t)\}_t$ that satisfy these constraints are said to be *feasible*. Note that the constraints depend on the specification of activity window curves, which are computed for given key event timetables. To examine the impact of alternative timetables, repeated LP runs are required, as already mentioned.

Resource Capacity Constraints: A resource allocation to OH_j may be feasible as far as the execution of OH_j is concerned, but we must take into account the competition for resources among overhauls. For each labor resource R_k, the total amount applied in the shipyard during time period t is constrained by the shop capacity $\bar{x}_k(t)$, which is assumed known. The resource

capacity constraints for each time period $t=1,\ldots,T$ are then given by

$$\sum_{j=1}^{J} x_k^j(t) \leqslant \bar{x}_k(t) \quad k=1,\ldots,K \ . \tag{4.1}$$

Objective

A set of resource allocations to overhauls is to be selected that utilizes the capacities (up to the time horizon T) most efficiently. More specifically, we seek feasible allocations $\{x_k^j(t)\}_t$ for all OH_j and R_k which satisfy (4.1) and maximize

$$\sum_{t=1}^{T} \alpha_t \sum_{k=1}^{K} \sum_{j=1}^{J} x_k^j(t) \ ,$$

where the coefficients $\alpha_1, \ldots, \alpha_T$ are still to be determined. Note that the resource application during period t for *all* labor types is weighted by α_t, since we assume that application quantities for different labor types are in comparable units and can be added together.

We define the efficient utilization of shipyard resources to mean that the sooner work is assigned resources, the better. Furthermore, the value that is placed on shifting a unit of resource use from a time period t to time $t-1$ is assumed to be independent of t. This implies that the coefficients α_t are linear in t and decrease with increasing t. They are set accordingly:

$$\alpha_t = T-t+1 \quad t=1,\ldots,T \ .$$

The resulting objective function is given by

$$\max \sum_{t=1}^{T} (T-t+1) \sum_{k=1}^{K} \sum_{j=1}^{J} x_k^j(t) \ . \tag{4.2}$$

The objective of applying resources as soon as possible is equivalent to operating the overhaul activities as "early" as possible, as we will now show. Let A_i be an activity within an overhaul OH_j, operating according to $OM_i=\{Z_i(t)\}_t$. Recall from Section 3.2.4 that the slack of A_i (the area between its window curves) is given by

$$\sigma_i = \sum_{t=S_i}^{F_i} Z_i^E(t) - Z_i^L(t) \ .$$

We define the *earliness* of A_i, denoted by P_i, as the proportion of the area between the window curves that lies below $Z_i(t)$:

$$P_i \equiv \frac{1}{\sigma_i} \sum_{t=S_i}^{F_i} Z_i(t) - Z_i^L(t) \ . \qquad (4.3)$$

If, for example, A_i operates early then $P_i=1$; if A_i operates late then $P_i=0$. Roughly speaking, P_i is the proportion of the slack of A_i that is not "used up" when A_i operates according to OM_i. Note that the earliness of A_i can also be interpreted as the weighted average relative progress of A_i during its operating window. That is, Equation 4.3 can be rewritten as

$$P_i = \sum_{t=S_i}^{F_i} \frac{\sigma_i(t)}{\sigma_i} p_i(t) \ ,$$

where $\sigma_i(t) \equiv Z_i^E(t) - Z_i^L(t)$ is the component of σ_i "accruing" from time period t.

It can be shown that the resource utilization objective (4.2) is equivalent to maximizing activity earliness, if the contribution of each activity is weighted appropriately. Let us consider the value of operating an activity A_i early. First, the larger its resource use (a_i), the greater the effect of early operation will be. Second, the larger its slack (σ_i), the greater is the difference between operating early or late. These two activity parameters are reasonable choices for weights on activity earliness. In fact, it turns out that the resulting objective, shown below, is identical to (4.2), up to a constant term:

$$\max \sum_{j=1}^{J} \sum_{A_i \in OH_j} a_i \sigma_i P_i \ .$$

This form of the resource utilization objective emphasizes the contribution of each aggregate activity.

At this point, our approach to aggregate planning becomes clear. We seek operating modes for all activities which are feasible, but which "pack" the shipyard with work as much as possible. In a sense, productivity is being maximized.

4.3. Updating Overhaul Data for LP Model

The resource allocation model described in the last section requires up-to-date information on all ongoing and future overhauls. If a key-op network exists for an overhaul, and is kept updated, then we can re-aggregate from it to reflect the current overhaul status. In practice, however, this may not be achievable. We therefore suggest updating methods at the aggregate level.

Labor Commitments

As an overhaul is executed, the status of each aggregate activity changes. In Section 4.1 we discussed labor commitments during an activity's operation. If the activity is about to start (or just started), its operation will be predetermined (i.e., its labor committed) for a portion of its window; if it is well underway then its operation may be completely predetermined. It is convenient to assume that the operation of an activity may be predetermined up to the end of either its first or last phase.

In order to transfer the relative progress of a partially predetermined activity to its successors, the value of one or both of its relative progress phase variables must be determined. For a given commitment of labor (converted to an operating mode), we approximate the value of a phase variable by the weighted average relative progress for time periods within the phase. This is analogous to activity earliness (see Section 4.2) being the average progress throughout the *whole* window.

Work Content Revisions

Now suppose that the work content is revised. If the revision applies uniformly during the activity's operation then its intensity can simply be rescaled. If a few, but significant, key-ops within the activity are revised then only a portion of the window may be affected. In this case the window curves must be modified in shape "locally" within the window.

Key Event Date Revisions

If the execution of an overhaul is delayed substantially, key event dates will need to be revised. Also, for future overhauls in the planning stage, alternative key event dates will be experimented with. In either case, these shifts in dates will cause the operation window of some activities to be shifted also.

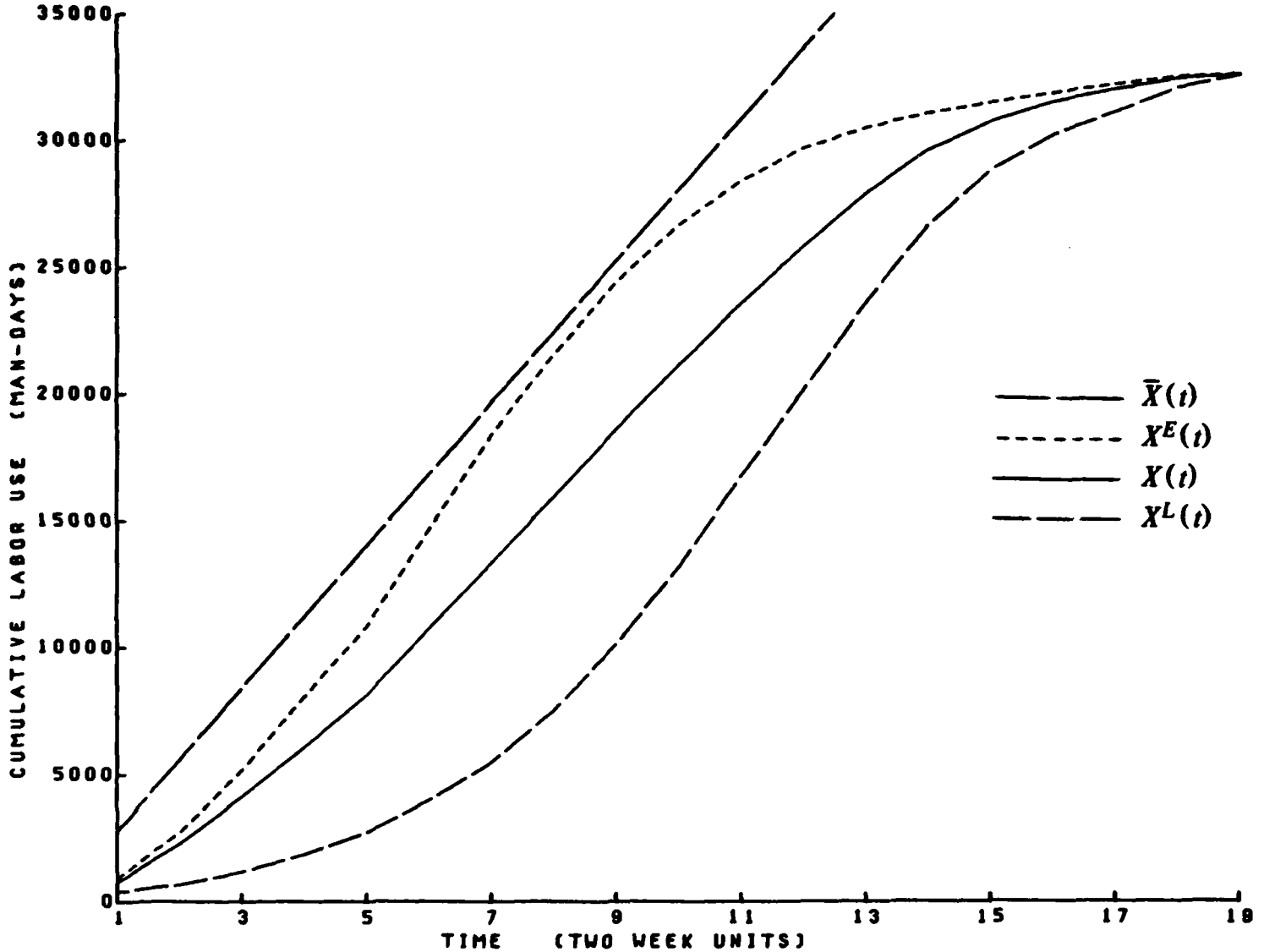


EXHIBIT 4.4

OVERHAUL DATA - CUMULATIVE RESOURCE USE

We consider the case where a key event is shifted forward by one time period, i.e., two weeks. (The shift backwards in analogous.) The early window curves of activities succeeding the key event should be shifted forward; the late curves of activities preceding it, backward. The shifts are then repeated (propagated) for activities that, respectively, succeed or precede these "neighbors" of the key event, etc.

Suppose an activity A whose early curve is shifted forward is one of several predecessors (not all of which are affected by the shift) of another activity B. If the slack of B is considerably less than that of A then the early curve of B is probably influenced more by predecessors other than A. In this case some other type of propagation of the shift in the key event date will be more appropriate, e.g., a stretching of the early curve of B.

4.4. Computational Examples

The LP model was applied to allocating resources to the overhaul described in Chapter 3; the most efficient overhaul labor use plan was computed for given labor shop capacities. The activity operation constraints consisted of 110 inequalities in 73 variables, plus upper bounds on the variables. (See Section 3.7.3.) For 20 time periods and 12 labor shops, 240 capacity constraints were added.

The obtained allocations are summarized by comparing *total* resource use with that derived from early and late overhaul execution without capacity constraints. Let $X(t)$ be the total amount of resource use up to time t in the optimal solution, so that

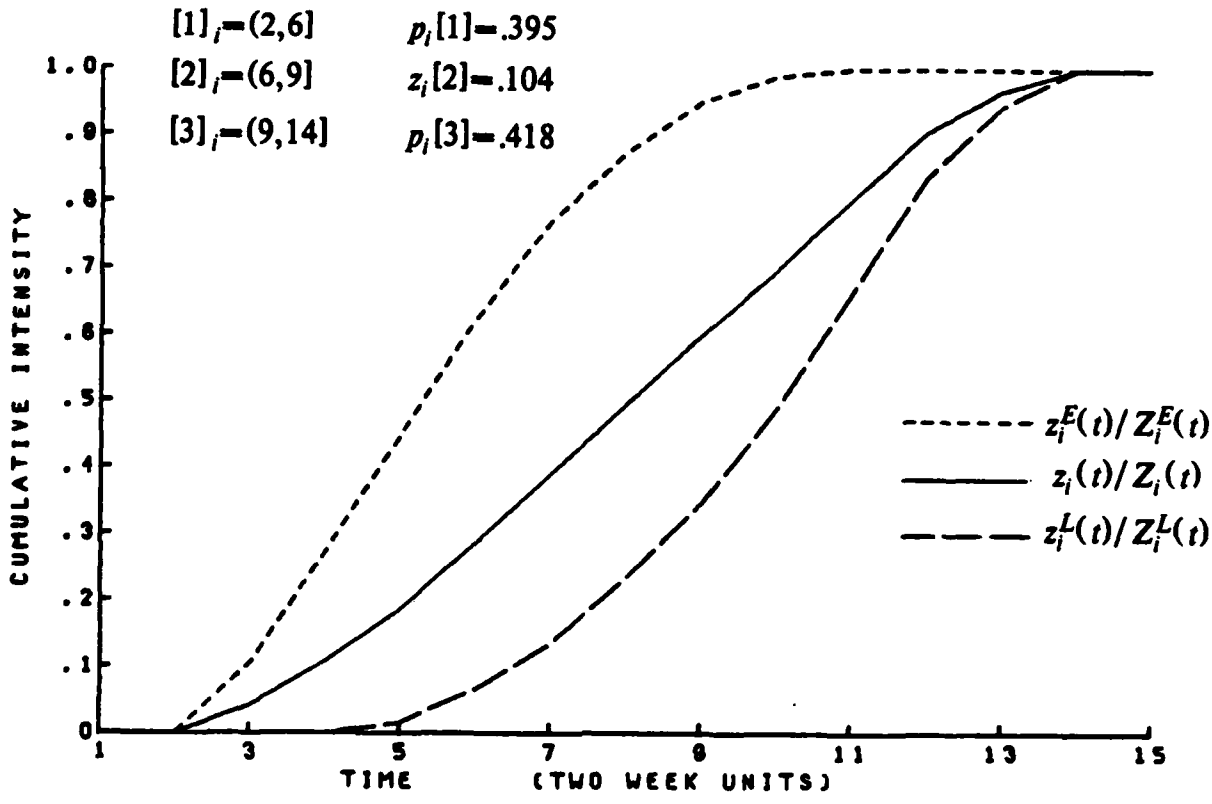
$$X(t) = \sum_{\tau=1}^{t} \sum_{k=1}^{K} x_k(\tau) \ .$$

Let $X^E(t)$ and $X^L(t)$ be the cumulative resource use for early and late unconstrained overhaul execution. Clearly,

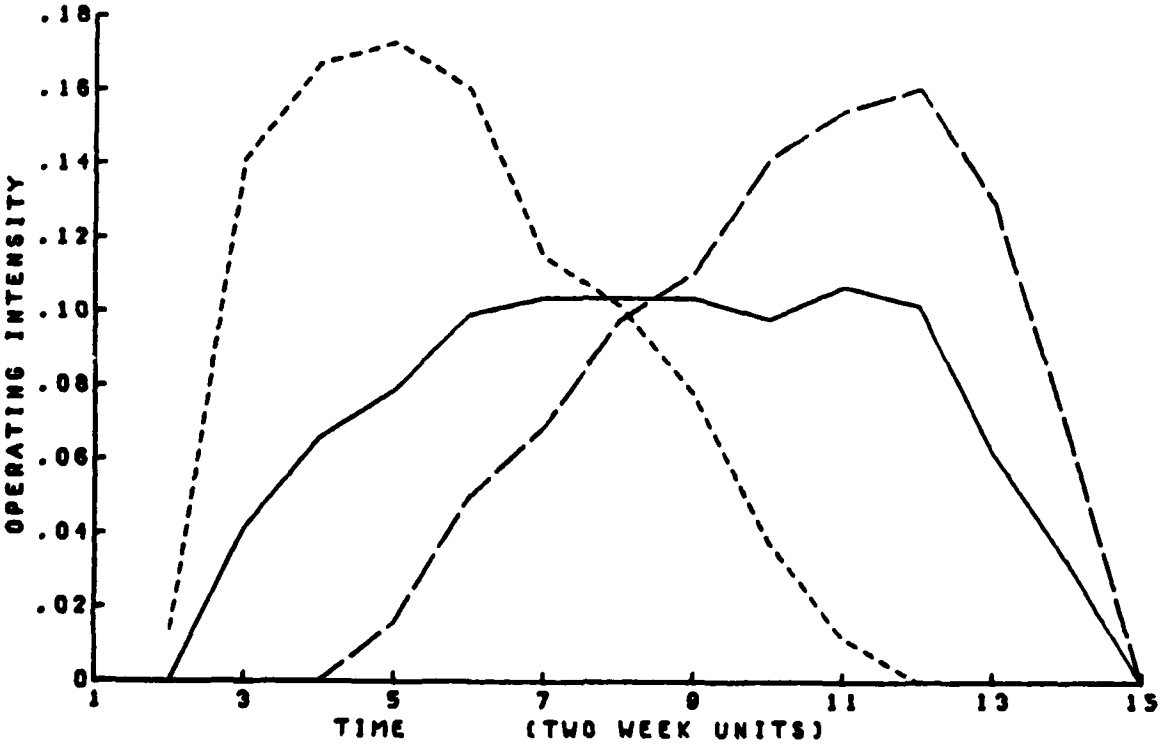
$$X^L(t) \leqslant X(t) \leqslant X^E(t) \qquad t=1,...,T \ .$$

If resource use were not constrained, the optimal solution would of course be early execution. However, labor shop capacities were set at about 60% of the peak loads arising from early overhaul execution, so that labor use must be leveled considerably.

Exhibit 4.4 shows the cumulative resource use curves. Note that the objective function (4.2) can be rewritten as



(a) Cumulative Intensity Curves



(b) Intensity Curves

EXHIBIT 4.5

OVERHAUL DATA - ACTIVITY OPERATING MODES

$$\max \sum_{t=1}^{T} X(t) ,$$

so that it can be interpreted as maximizing the area under the curve $X(t)$. Let $\bar{X}(t) = \sum_{\tau=1}^{t} \sum_{k=1}^{K} \bar{x}_k(\tau)$ be the cumulative available capacity, also shown in Exhibit 4.4. The slope of $X^E(t)$ and $X^L(t)$ exceed the slope of $\bar{X}(t)$ at some time points, illustrating that they would violate the capacity constraints. Also, while $X(t)$ does not seem to reach the slope of $\bar{X}(t)$ anywhere, this is due to the fact that labor application from *each* shop is constrained, not only total labor use.

The capacities, as they were set, still allowed considerable flexibility in overhaul execution. All cumulative resource curves $X(t)$ will lie within the area between the extremes $X^E(t)$ and $X^L(t)$. In the optimal solution the curve lies above 63% of that area. As an experiment, the objective function was *minimized* (the later that work is assigned resources, the better). The resulting cumulative resource curve was above only 35% of the area between the extremes. The flexibility of executing the overhaul is thus equivalent to 63-35% = 28% of the area.

Exhibit 4.5 shows an example of a three phase activity operating mode obtained from the LP, together with its early and late operation extremes. (This is activity A_{27}; see Exhibit 3.9.) Note that constant relative progress within a phase does not imply monotone behavior of operating intensity, as shown in the last phase.

Let us consider a more realistic problem size. We assume a constraint set per overhaul of 150 inequalities in 100 variables. While some of the overhauls will be partially completed, suppose the equivalent of 15 whole overhauls are to be analyzed. Setting a time horizon of two years (50 time periods) with 12 labor shops, the total problem size is $(150)(15)+(50)(12)=2850$ inequalities in $(100)(15)=1500$ variables, which is readily handled by LP computer packages. We see that, computationally, the resource allocation model is feasible.

REFERENCES

[1] Cooper, D. F., "Heuristics for Scheduling Resource Constrained Projects: An Experimental Investigation," *Management Science,* Vol. 22, No. 11, p. 1187 (1976).

[2] Elmaghraby, S.E., *Activity Networks: Project Planning and Control by Network Models,* John Wiley & Sons, New York (1977).

[3] Harris, R.B., *Precedence and Arrow Networking Techniques for Construction,* John Wiley & Sons, New York (1978).

[4] Herroelen, W.S., "Resource Constrained Project Scheduling - the State of the Art," *Operational Research Quarterly,* Vol. 23, No. 3, p. 261 (1972).

[5] Leachman, R.C., "Aspects of Dynamic Production Planning," *Report ORC 79-8,* Operations Research Center, University of California, Berkeley (1979).

[6] Leachman, R.C., "Production Planning for Multi-Resource Network Systems," to appear in *Naval Research Logistics Quarterly* (1982).

[7] Leachman, R.C. and J. Boysen, "Multiple Resource Smoothing in Shipyard Planning," *Report ORC 81-15,* Operations Research Center, University of California, Berkeley (1981).

[8] Moder, J.J. and C.R. Phillips, *Project Management with CPM and PERT,* Second Edition, Van Nostrand and Reinhold, New York (1970).

[9] Patterson, J.H., "Alternate Methods of Project Scheduling with Limited Resources," *Naval Research Logistics Quarterly,* Vol. 20, No. 4, p. 767 (1973).

[10] Shephard, R.W., R. Al-Ayat and R.C. Leachman, "Shipbuilding Production Function: An Example of a Dynamic Production Function," *Quantitative Wirtschaftsforschung Festchrift Volume (Wilhelm Krelle),* J.B.C. Mohr (ed.), Tubingen (1977).

[11] Shephard, R.W. and K.-T. Mak, "Simulation of Ship Overhaul Project Network," *Report ORC 82-3,* Operations Research Center, University of California, Berkeley (1982).

tutorial information. Tentative key event timetables are then considered; in particular, trial dates are established for important restraining events such as docking and undocking.

END

FILMED

1-83

DTIC